RC22933 (W0310-060) October 9, 2003
Computer Science

Jazzing up Eclipse with Collaborative Tools

Li-Te Cheng, Susanne Hupfer, Steven Ross, John Patterson
IBM Research Division
Cambridge, MA 02142

Jazzing up Eclipse with Collaborative Tools

Li-Te Cheng, Susanne Hupfer, Steven Ross, John Patterson
IBM Research
Cambridge, MA 02142
{li-te_cheng,shupfer,steven_ross,john_patterson}@us.ibm.com

Abstract

Collaboration is an integral part of software development, occurring through tools inside and outside the IDE. This paper presents an overview of the Jazz project, which seeks to integrate collaborative capabilities into the Eclipse IDE, enabling small teams of software developers to work together more productively.

1 Introduction

Software development is a collaborative process, where teams of developers work together to design, solve problems, and produce quality code [2]. In addition to live face-to-face interactions, developers use a variety of collaborative tools in their everyday activities inside and outside of their IDE. These can include formal tools such as source control and bug tracking systems, and ad hoc ones such as email and instant messaging. As development teams become increasingly distributed (e.g. outsourced/off-shore development, open source projects), there is a need for computer-based collaborative tools to support structured and unstructured communication and coordination of work. Integrating such tools into the IDE, and enabling them with awareness of development processes and artifacts, may help reduce context switches between tools inside and outside the IDE and make the connection between development and collaboration more seamless [2].

The Eclipse platform [12] has much to offer to support an integrated collaborative software development environment. The tight integration of many different development tools, ranging from editors to source control, makes Eclipse the "desktop" for the developer. The platform has various extension points for others to contribute their own add-ons, and to gather contextual information. Thus, there is an opportunity to insert additional tools that not only support collaboration but also take into account the developer's current context, and appear fully integrated into the IDE.

This paper presents an overview of the Jazz project, which aims to enable small teams of developers to work together more effectively by integrating collaborative capabilities into the Eclipse IDE. We begin with a review of related work, present the metaphor motivating our project, and then describe its key features in detail. We conclude with future directions for this work.

2 Related Work

Booch and Brown identify the motivation and requirements for collaborative development environments, survey their current state in the marketplace as well as some of their research heritage, and outline future directions [2]. They point out that a rich collaborative development environment arises from the collection of many seemingly simple collaborative components to support coordination, collaboration, and community building. They also state that IDEs equipped with team-centric features are a step up from those merely augmented with some collaborative support.

Non-collocated software development teams face considerable collaboration challenges. Herbsleb and Grinter studied such teams [16], finding that while architecture, plans, and processes were important, so was support for ad hoc, flexible communication. They recommended using tools for finding organizational information, supporting spontaneous meetings across different sites, and maintaining awareness about other people's availability. In addition, a study by de Souza, Redmiles, and Dourish suggests that awareness of co-workers' activities around shared code is also important [4].

Various systems have been built to address some of these issues. A number of them focus on enhancing the source control system with fine-grained change mechanisms or visualizations of code changes by team members over time [17,18,20]. A notable early system is Flesce [9], which featured a shared testing environment, shared debugger, shared editor, audio conferencing, audio annotation, and shared code reviewing.

Source control systems like ClearCase and CVS support basic code awareness by sending email when specified files are changed [8]. And Eclipse itself has a number of built-in collaborative capabilities. Features like Quick Diff and the CVS Annotation View provide awareness of code changes with respect to the code repository and indicate who is responsible for the changes [11].

There are also a growing number of Eclipse plugins and projects supporting various facets of group collaboration. The Eclipse Wiki plugin allows creation of personal and group hyperlinked documents [22]. Sangam enables users to chat around a shared editor to support pair programming efforts [19]. Quickshare provides ad hoc, lightweight code sharing among developers [10]. The Koi project is building a collaborative infrastructure for Eclipse applications [13]. The Gild project is enabling Eclipse for collaborative learning [15]. Hipikat ties in shared software and collaboration artifacts to aid developers [17].

Three Eclipse-related projects, in particular, strongly focus on the integration of many collaborative features with software development practices and processes. CodeBeamer is a product directed at team collaboration, and has plugins for shared task management and bug tracking, with plans to integrate group discussion, project management, and document sharing [6]. The Composent Plugin provides collaborative capabilities such as group chat, file sharing, co-browsing, application sharing, remote program launching, and awareness of team members' currently open tasks, files, and UI selections [3]. The Stellation project introduces fine-grained source control and uses the notion of activities to simplify collaboration and provide awareness of others' changes. It features lightweight activity authoring, and file associations, enabling developers to manage relevant work, notify the team of their current work, and be informed of changes pertaining to their own activities, and also provides a context for persistent conversations [14].

3 Jazz Metaphor

Jazz is based on the metaphor of an "open office" approach to application development, as advocated by approaches like Extreme Programming [1]. In such a setting, a team of developers works in close proximity. Within the office, each developer will have his or her own workstation, while elsewhere in the office will be space for team meetings, shared whiteboards, schedule information, etc. A key aspect of this setting is team awareness. Even while focusing on their personal work, all team members have a sense of what is going on elsewhere in the room, who is talking to whom, what others are working on, etc. Communication among team members is also facilitated in this environment, whether in the form of a question shouted out to the team in general, or in calling a colleague over to consult on an issue at a particular workstation.

We seek to extend the Eclipse Platform to capture these features of an open office environment. Our goal is to elevate the team, a small egalitarian group of people with common goals and implicit trust in one another, to a first-class object in the development environment, and to facilitate awareness, communication, and coordination between the members of the team. Our focus is on the core team of code developers, not the extended team, which includes management and support personnel, nor the extended community that includes the team. Our expectation is that these teams will tend to have from two to ten members, though we are not planning on enforcing any hard limits on team size.

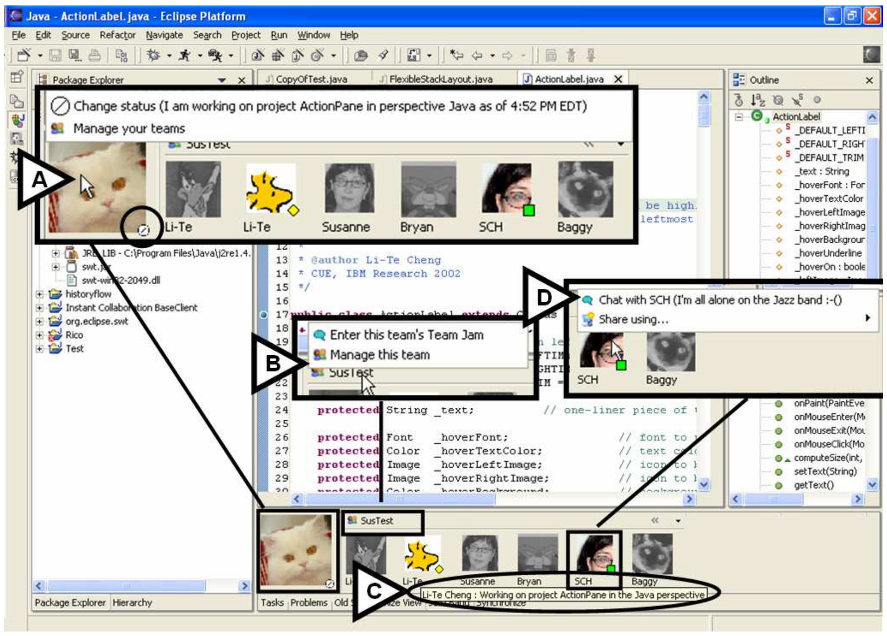
4 Key Features

The Jazz project focuses on a specific set of collaborative features to support the core team of developers: the "Jazz Band", chat, screen sharing, and "Concert Awareness".

The most visible aspect of the Jazz enhancements to Eclipse is the Jazz Band, a viewpart that provides the user with peripheral awareness of the other team members (see Figure 1). It shows the user and lists the teams the user belongs to, as well as members of those teams.

Teams in Jazz are meant to be small, informal, ad hoc, and invitation-based: anyone can create a team, and add anyone to a team. Any team member can subsequently add or remove team members, although dropping oneself from a team removes one's team privileges. Unlike a typical instant messaging buddy list, the Jazz Band is a communal list: team state is shared and synchronous, so any changes to the team immediately show up on all team members' Jazz Bands.

As shown in Figure 1A, each team member in the Jazz Band is represented by a portrait, and is decorated by an online status icon indicating whether the person is online, away, or busy.

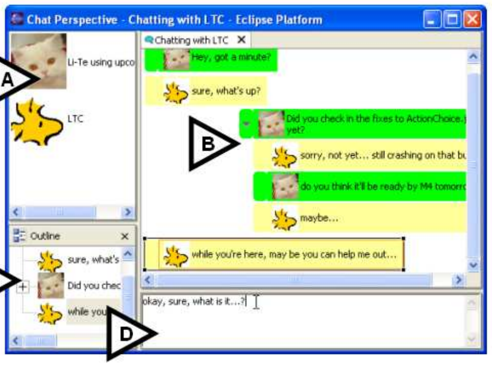

Java - ActionLabel.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
CopyOfTest.java
FlexibleStackLayout.java
ActionLabel.java
Outline
Change status (I am working on project ActionPane in perspective Java as of 4:52 PM EDT)
Manage your teams
SusTest
Li-Te
Li-Te
Susanne
Bryan
SCH
Baggy
A
B
C
D
Enter this team's Team Jam
Manage this team
Chat with SCH (I'm all alone on the Jazz band :-()
Share using...
historyflow
Instant Collaboration BaseClient
org.eclipse.swt
Rico
Test
Tasks
Problems
Package Explorer
Hierarchy
Li-Te Cheng : Working on project ActionPane in the Java perspective

```
 * @author Li-Te Cheng
 * CUE, IBM Research 2002
 */

    protected String _text;        // one-liner piece of

    protected Font    _hoverFont;        // font to
    protected Color   _hoverTextColor;   // text co
    protected Image   _hoverLeftImage;   // icon to
    protected Image   _hoverRightImage;  // icon to
```

Figure 1: The Jazz Band: (A) pop-up menu to set the user's status message and manage all teams; on the bottom right of each portrait is an IM status; offline team members have dark grayscale portraits, (B) people are grouped in user-defined teams; each team has a pop-up menu to enter the team's "Team Jam, (C) hovering over portraits reveal their online status messages (an option can be set to overlay the messages on the portraits), (D) pop-up menu for a team member reveals options to start a chat and a screen share



Hey, got a minute?
sure, what's up?
Did you check in the fixes to ActionChoice.j yet?
sorry, not yet... still crashing on that bu
do you think it'll be ready by M4 tomorro
maybe...
while you're here, may be you can help me out...
sure, what's
Did you chec
while you
okay, sure, what is it...?

Figure 2: Chat: (A) list of who is in the chat right now, (B) chat transcript with moveable items, threading, and zooming (C) outline view, (D) text input area

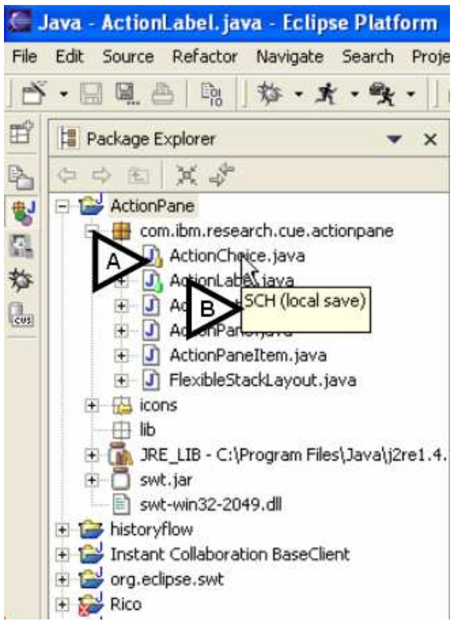

Java - ActionLabel.java - Eclipse Platform
File Edit Source Refactor Navigate Search Proje
Package Explorer
ActionPane
com.ibm.research.cue.actionpane
ActionChoice.java
ActionLabel.java
SCH (local save)
ActionPaneItem.java
FlexibleStackLayout.java
icons
lib
JRE_LIB - C:\Program Files\Java\j2re1.4.
swt.jar
swt-win32-2049.dll
historyflow
Instant Collaboration BaseClient
org.eclipse.swt
Rico
A
B

Figure 3: Concert Awareness: (A) files with local changes are decorated with colored icons on the bottom right of each file icon, (B) tooltip over a file reveals who made the local change

Offline members do not have a decorator icon and are represented by a dark grayscale version of their portrait image. Thus at a glance, one can gauge whether a co-worker is working in the Jazz-enhanced Eclipse environment or not. The portrait's tooltip reveals the individual's personal status message (Figure 1C). Alternatively, the user can set a preference to overlay the status messages.

A difference from status messages in typical instant messaging applications is the ability to "scat" in Jazz. Scats (an acronym for "status command and text") are optional macros the user can insert into a status message. In addition to the date and time information, Jazz provides scats to reveal Eclipse-specific information, such as the user's currently active project, perspective, editor, and file.

Clicking on a portrait brings up a pop-up menu with a variety of choices (Figure 1D), notably the option to initiate a chat. At a first glance, chatting in Jazz is similar to other chat systems. Figure 2 shows a Jazz chat window, which includes: a list of active participants (Figure 2A), a chat transcript area (Figure 2B), an outline version of the chat transcript (Figure 2C), and an area to enter text (Figure 2D). By default, the user can type messages into the text input area and the chat transcript stacks messages in a list-like layout. Unlike a typical chat, however, every item in the chat transcript area can be moved, and can be threaded. For example, in Figure 2B, the middle of the conversation was moved to the right to highlight that piece as a different thread of conversation. Messages will continue to scroll downwards like a normal chat, but the user is free to move out messages of interest. The layout is personal – position information is not shared with others. When a message is actively selected by the user, typing will nest the next message as a response to the selected message. This threading relationship is shared among chat participants. Threading allows users to define newsgroup-like discussion threads which can be collapsed to reduce clutter. The outline view in Figure 2C provides an ordered tree view of all chat messages, so it can be used to jump to specific messages that are out of view or collapsed in a thread. The transcript area is also zoomable: right clicking will bring up a context menu to zoom in or zoom out, so the user can navigate through large conversations. Chats in Jazz can also be saved for later review.

Clicking on a team name (Figure 1B) brings up a pop-up menu with access to the "Team Jam," a team-centric discussion board. The Team Jam user interface is similar to the chat interface discussed earlier, but with two major differences. First, messages can include non-chat messages, such as links to transcripts of older chats, and team event notifications, such as code check-ins and check-outs from source control. Second, all positioning information is shared, and thus the Team Jam looks like a shared whiteboard of moveable message items.

Beyond asynchronous discussions in the Team Jam and synchronous chat discussions, Jazz allows a user to share his/her screen with other team members to support activities such as joint debugging, code walkthroughs, and general consulting. The user can initiate a screen sharing session by clicking on a portrait and selecting "Share using…" (Figure 1D), automating steps like starting the screen sharing service and alerting the other party about the screen sharing session. Jazz currently supports sharing using TightVNC [21].

In addition to people-centered awareness, the Jazz project provides resource-centered awareness, which we call "Concert Awareness", through extensions to the Eclipse package explorer (see Figure 3). Concert Awareness adds colored resource decorators (Figure 3A) to indicate what others are doing with their local copies of the files (e.g. indicating that the file was locally saved but not checked back in, and if the file is currently in focus and being changed at this very moment). Tooltips on such resources reveal who is responsible for these changes (Figure 3B).

6 Future Directions

The goal of the Jazz-enhanced Eclipse environment is to provide easy, in-context access to as much of the team information as possible, without hindering the user's work unnecessarily. Annotations, questions, answers, chats, screen shares, and file reservations are all stored in the Team Jam, and are also accessible from editor markers, personal icons on the Jazz Band, and the Package Explorer. We will be experimenting with cross-linking these artifacts, so that, for example, file reservations will specify an associated task, tasks can refer to locations in the code, code can contain references to annotations, questions, and answers, etc. The goal is to produce a web of related information that knits the members of the team into a well-coordinated, productive working group.

We have also conducted a series of interviews with professional developers discussing the management of interdependencies across teams and gathered their reactions to Jazz. We plan to publish the results of this study and use them to inform the design of our next iteration of Jazz.

Acknowledgements

We would like to thank our interns, Bryan Clark and Cleidson de Souza, for their significant contributions to the Jazz project.

About the Authors

The authors are members of IBM Research in the Collaborative User Experience group at Cambridge, Massachusetts. Their expertise center around the technologies, user interfaces, and social aspects behind groupware. They can be reached via email at li-te_cheng@us.ibm.com, shupfer @us.ibm.com, steven_ross@us.ibm.com, and john_patterson@us.ibm.com.

References

[1] K. Beck, *Extreme Programming Explained: Embrace Change*, Addison-Wesley, 1999.

[2] G. Booch and A, Brown, Collaborative Development Environments, in *Advances in Computers Vol. 59*, Academic Press, August 2003.

[3] Composent Eclipse Plugin, http://composent.com/code/eclipsesite/

[4] C. de Souza, D. Redmiles, and P. Dourish, "Breaking the Code", Moving between Private and Public Work in Collaborative Software Development, to appear in *Proc. ACM Group*, Sanibel Island, USA, November, 2003.

[6] CodeBeamer, http://www.intland.com/

[7] Collaborative Integrated Development Environment, http://hci.usask.ca/projects/collide.xml

[8] cvshome.org, Telling CVS to notify you, http://www.cvshome.org/docs/manual/current/cvs_10.html#SEC91

[9] P. Dewan and J. Riedl, Toward Computer-Supported Concurrent Software Engineering, *IEEE Computer*, pages 17-27, January 1993.

[10] Eclipse - Collaborative Development Tools Project, http://www.scs.carleton.ca/~skaegi/cdt/index.html

[11] Eclipse.org, Eclipse 3.0 M2 – New and Noteworthy, http://download.eclipse.org/downloads/drops/S-3.0M2-200307181617/eclipse-news-M2.html

[12] Eclipse.org, Eclipse Platform Subproject, http://www.eclipse.org/platform

[13] Eclipse.org, Koi: A Collaborative Infrastructure for Eclipse Tools, http://www.eclipse.org/koi

[14] Eclipse.org, Stellation: Advanced Software Configuration Management, http://www.eclipse.org/stellation

[15] GILD: Groupware enabled Integrated Learning and Development, http://gild. cs.uvic.ca

[16] J. Herbsleb and R. Grinter, Architectures, Coordination, and Distance: Conway's Law and Beyond, *IEEE Software*, pages 63-70, September/October 1999.

[17] Hipikat: Recommending Useful Software Artifacts, http://www.cs.ubc.ca/labs/spl/projects/hipikat/

[18] B. Magnusson and R. Guerraoui, Support for Collaborative Object-Oriented Development, in *Proc. Int. Symp. On Parallel and Distributed Computing Systems*, Dijon, France, 1996.

[19] Sangam Project, http://sangam.sourceforge.net

[20] A. Sarma, Z. Noroozi, and A. van der Hoek, Palantír: Raising Awareness Among Configuration Management Workspaces, in *Proc. 25th Int'l. Conf. on Software Engineering*, pages 444-454, Portland, USA, May 2003.

[21] TightVNC, http://www. tightvnc.com/

[22] Wiki Editor Plugin, http://www.teaminabox.co.uk/downloads/wiki