ABSTRACT
Pair programming has proven to be a useful technique for developing high quality code while sharing knowledge throughout a team. Rapid global dispersion of software development teams, however, makes co-located pair programming a challenge, motivating the need for development tools tailored specifically for distributed pair programming. Previously, the Sangam Eclipse plug-in was developed to support distributed pair programming. More recently, the Jazz collaborative software development platform was built to support team communication and the sharing of life-cycle resources and to integrate a variety of disparate tools used by team members. We have ported Sangam to the Jazz platform to enable teams to pair program within their integrated team environment. In this paper, we describe Jazz Sangam, highlight the choices that lead to Sangam’s current design, and discuss how Jazz Sangam can improve the distributed pair programming experience.

Categories and Subject Descriptors
D.2.9 [Software Engineering]: Management—Programming Teams

General Terms
Collaboration

Keywords
Distributed Pair Programming, Collaboration

1. INTRODUCTION
Pair programming has proven to be a useful collaboration technique for developing high quality code while sharing knowledge throughout a team[12]. But today, many organizations have multiple locations, leading to a significant change in the way people work and communicate. Open source development depends on developers working remotely. Such distributed teams cannot always be co-located to collaborate, motivating the need for distributed collaboration tools.

Jazz is a collaborative software development platform from IBM that brings several features like source control, defect management, team process, instant messaging and a web portal together in a single package. We developed a Jazz feature, Jazz Sangam, to provide distributed pair programming support for the Jazz platform. Jazz Sangam is extended from Sangam[10], an Eclipse plug-in for distributed pair programming.

Based on event-driven design, Sangam offers a synchronized workspace for Eclipse Java developers with minimal bandwidth requirements. Since Sangam does not have a central code repository, users must have local copies of code when they like to pair program. Sangam has a constraint that the code on all workspaces is the same/consistent before starting the pair-programming session. This constraint is necessary since only changes are transmitted from the driver to the navigator. Sangam has no built-in functionality to verify code consistency, so we have extended Sangam for Jazz platform to benefit from Jazz’s integrated features like Jazz Source Control, Work Items and Chat View. In this paper, we describe how the Jazz-Sangam combination was designed to provide support for distributed pair programming within a team development environment.

1 www.jazz.net
3. RELATED WORK

Earlier research\[12\][7] indicated that through pair programming, higher quality code could be produced in a similar amount of time when compared with solo programming. Correspondingly, Baheti et al\[1\] studied the quality and productivity of distributed pair programmers. Their study with student teams across North Carolina State University and University of North Carolina-Chapel Hill using tools like VNC, Microsoft NetMeeting and instant messengers indicated that distributed pair programming was as effective as colocated pair programming.

Nick Flor stressed that supporting awareness through cross-workspace, visual and audio channels is critical to reproduce and improve any distributed collaboration experience\[3\]. Ishii et al\[5\] found that collaborating developers communicate by pointing at screen using hand or mouse cursor, called gesture awareness, is an important form of non-verbal communication. Gesture awareness can be incorporated in distributed pair programming using telepointers \[4\]. At times, collaborators find it necessary to see where their partners look. The term for this is gaze awareness. Efforts to support gesture and gaze awareness in collaboration tools were successful \[5\][6]. Jazz Sangam, however, does not include support for gesture and gaze awareness; rather, it focuses on workspace awareness, which is the knowledge about shared artifacts, co-workers, activities and extension of the shared workspace\[9\]. In the next section, we describe how Sangam takes advantage of a collaborative team platform like Jazz.

4. JAZZ AND SANGAM

This section describes Jazz and Sangam and how they enhance each other.

4.1 Sangam

Sangam was created to support distributed pair programming functionality on Eclipse, even on slower network connections. Based on an event-driven design, the actions of the driver in the Eclipse IDE are intercepted by the plug-in and transmitted to the navigator’s end to automatically perform the same task. The result is a synchronized workspace updated in real time.

The plug-in was designed as three modules: an event interceptor, a message server, and an event reproducer. The events are intercepted and broadcasted as encoded messages by the event interceptor, while the event reproducer parses the messages and performs the appropriate actions at the navigator’s end. To closely integrate the message handling with other modules, a plug-in was developed on the basis of a lightweight message server called Kizna SyncShare\[5\].

Figure 1 shows a Jazz workspace after installing Sangam. When two developers want to start pair programming, they first login to SyncShare, setting up a connection between the two developers. After obtaining consistent code resources and opening them in the Sangam Editor, the developer wishing to ‘drive’ clicks the “Start Driving” button (right button on toolbar). This “Start Driving” button is then disabled for other developers, while it is toggled to a “Stop Driving” button for the driver. The developers can switch roles only after the driver clicks on “Stop Driving” button. The developers can launch a Java application or JUnit test together by using Sangam Launchers.

One requirement for Sangam to work properly is that the code between the pair programmers needs to be consistent before starting to pair-program; otherwise erroneous results

\[7\]http://sourceforge.net/projects/syncshare/
are seen in the source code. Users might have different versions of code on their workspaces. Jazz Sangam solves this issue because both partners can start with the same code checked out of the Jazz source control repository.

4.2 Jazz

Jazz is an integrated development platform by IBM, based on client-server architecture with an emphasis on many real-time collaborative technologies. Jazz is a process and team-aware platform, which automates both individual and team workflows. Developers can use remote clients with integrated Jazz support to access code from the Jazz server and develop collaboratively. The Jazz architecture is based on open web standards and is highly extensible. In this paper, we will describe three major features of the Jazz platform that integrate well with the Sangam plug-in: Jazz Source Control, the Chat view, and Work Items.

Similar to version control systems such as Subversion, Jazz implements its own version control system called as Jazz Source Control. Additionally, Jazz keeps track of local changes of a user, called "Change Sets". The idea is that every change in the project must be a logical change associated with a description, task, defect, or enhancement. A developer can manage several change sets locally, and using the "Pending Changes" can view the changes sets of their teammates in real-time.

One of the central features of Jazz is its defect tracking system, called "Work Items". Similar to Bugzilla, work items can be opened and resolved from any Jazz-enabled client. They can be associated with change sets, patches, screen captures, and many other resources related to task management. In section 5.2 we suggest an enhancement to Jazz Sangam based on Work Items that can reduce manual work required for Jazz users to pair program.

Another collaborative feature of Jazz is an instant message client within the development environment. The Chat view can be configured to work with Google talk, an OpenFire server, or several other Jabber protocols. Chat conversations among teammates can be attached to work items or archived for later review. Below, we describe how these Jazz features enhance distributed pair programming.

4.3 Course Integration of Jazz

In the Spring 2008 semester, we used the Jazz development platform as an extra credit option in an undergraduate Software Engineering team project at North Carolina State University. The instructors provided the students with a tutorial for deploying the team project in Jazz, and an introduction to a few of the Jazz features. Students were asked to explore the system’s features and to keep track of which features of Jazz they used the most and found most helpful.

The two most popular features mentioned involved the Jazz version control: a "Pending Changes" window that allows students to watch which files are being modified by their teammates in real-time, and a pop-up notification that shows up when conflicting changes are submitted by others. Students reported that this pop-up provided important communication in their development: eight out of eleven students (72% of the students who used Jazz), reported this pop-up notification as one of their favorite features.

Although a pop-up is only slightly different than constantly checking for new changes, many students reported that this real-time feature allowed for easier coordination during times of distributed development. That the students responded so positively to the real-time distributed development features of Jazz indicates that a feature like Sangam would integrate well with the Jazz platform.

6http://subversion.tigris.org/
7http://www.bugzilla.org/
8http://www.google.com/talk/
9http://www.igniterealtime.org/projects/openfire/
10www.jabber.org
4.4 Jazz Sangam

To use Sangam, code needs to be consistent among the pair programmers. Using Jazz Source Control, developers can simply check the versions of their individual codes using "Pending Changes" feature before starting pair programming and can be sure that their code is consistent. Alternatively, the developers can use a separate repository for pair programming from which they can access the same versions of code. Also, if the navigator accidentally enters anything into his editor, the code will be inconsistent from that point (since Sangam transmits edits and not the file contents). In such a case, the navigator can use "Pending Changes" view to find the differences between his workspace and driver’s workspace and use auto-resolve to fix the inconsistencies.

The Jazz-Sangam relationship is mutually beneficial. Jazz does not currently provide a distributed pair programming feature like Sangam. The context-sensitive features of Jazz further enhances distributed pair programming functionality through a seamless integration with Sangam. We suggest few possible enhancements below.

5. FUTURE ENHANCEMENTS

Perhaps the most notable advantage of the Jazz platform is its integration of many different technologies. The most significant area of future enhancement for Jazz, therefore, is a more seamless integration of Sangam with Jazz features.

5.1 Network Integration

The configuration process for Jazz Sangam can be integrated to use additional Jazz resources. Experience has shown that users will rarely use a feature if they perceive it to have complex configuration. Hence, the next step is to integrate SyncShare server with Jazz server so that users do not have to configure IP addresses for SyncShare when they want to start pair programming.

5.2 Context-Sensitive Interface Enhancements

As reported by our students, one of the main advantages of Jazz is using context-sensitive features. Chat windows, source control, work items, unit testing, and code editing are all immediately accessible from the workbench and contain relevant links to each other. The current version of Jazz Sangam is relatively isolated from the rest of the Jazz features. Creating seamless, context-sensitive interface integrations would likely lead to more usage. An example of this integration would be to create links from a Work Item, attached with relevant code, screen captures etc, to the Sangam editor so that two users can start pair programming quickly, avoiding the process of connecting to a repository and downloading the relevant code.

5.3 Ensuring Synchronization

Although the value of Sangam is in its low-bandwidth design, two developers must pair-program code that is perfectly synchronized. Without source control, ensuring that both developers have the same code can be tedious and difficult. By leveraging Jazz source control, the Sangam plugin would automatically check that the files opened in the Sangam editor are exactly the same for both developers.

Distributed pair programming is an effective way to collaborate especially with context-sensitive tools like Jazz and Sangam. Seamless integration of Jazz Sangam with Jazz will further enhance collaboration. From our experience in the course integration of Jazz, we can find that the real-time feedback like pop-up notifications and context-sensitive help which reduces manual configurations are among the preferred enhancements to any collaboration tool. We believe that the value of Jazz could be increased by a tool like Sangam, and look forward to expanding our plug-in to be more tightly integrated into Jazz.

7. ACKNOWLEDGEMENTS

The development of the Jazz Sangam was funded by an IBM™Jazz Faculty Grant.

8. REFERENCES