On Preparing Students for Distributed Software Development with a Synchronous, Collaborative Development Platform

Andrew Meneely and Laurie Williams
North Carolina State University
Raleigh, NC, USA
{apmeneel, lawilli3}@ncsu.edu

ABSTRACT

Working remotely is becoming the norm for both professionals and students alike. Software development has become a global industry due to outsourcing, teleworking, flex time, and companies' desire to use the best and/or most economical talent regardless of where that talent is located. Professionals are not alone because students usually work from home despite having sufficient resources on campus. In this paper we share our experiences from using Jazz, a synchronous, collaborative development platform, with our inevitably distributed software engineering students. Eleven students optionally used the tool while working on a five-week team project. Students primarily used the version control, chat, and work items features in Jazz. We collected their reactions in retrospective essays and found that all Jazz students supported using Jazz in future semesters of the course. We also examined grade differences and found that the students who used Jazz were more successful than those who did not use Jazz.

Categories and Subject Descriptors

K.3.2 [**Computing Milieux**]: Computing and Information Sciences Education – *computer science education*

General Terms

Human Factors

Keywords

distributed development, collaboration, software engineering,

1. INTRODUCTION

Working remotely is becoming the norm for both professionals and students alike. Software development has become a globally distributed industry [4, 7] due to outsourcing, teleworking, flex time, and companies' desire to use the best and/or most economical talent regardless of where that talent is located. By some accounts[4], 80 percent of software projects are now global. Students will often choose to work in the comfort of their home without taking the time to travel back to the laboratory despite having access to the most well equipped computer laboratory on campus. Fortunately, working remotely will prepare students for the distributed work arrangement that is likely to await them in their professional career.

Educators are taking explicit action to prepare students to be productive when working remotely via structuring courses shared by students in multiple universities [2, 6]. Key success factors for global software development include a focus on information management, clear and accessible communication and the use of a flexible development infrastructure [4] for both asynchronous and synchronous development. Academic courses, therefore, need to provide development platforms for students to support these key success factors while also supporting both asynchronous and synchronous development.

In this paper we share our experiences with students using Jazz[1], a synchronous, collaborative development platform, produced by IBM, which provides such a supportive development platform. We decided to use Jazz with our inevitably distributed teams for four reasons:

- Jazz integrates support for team development in one platform. Teaching multiple tools can be a distraction from the course objectives.
- Due to its integration, features such as version control and defect reporting require less effort and, therefore, are more likely to be used by students beyond the minimum use specified in an assignment.
- Students gain added benefits when working at the same time as other teammates whether this synchronous work was explicitly planned or unplanned and opportunistic.
- Jazz is built upon Eclipse[2], a development environment familiar to students.

We used Jazz with third- and fourth- year students in a software engineering course. Students had the option to use the tool over our usual environment (Eclipse, Subversion version control, Bugzilla defect tracking)[3] while working on the five-week team project. We collected their reactions about the use of Jazz via written course retrospectives and examined student performance by analyzing grades.

[1] http://jazz.net
[2] http://eclipse.org
[3] http://subversion.tigris.org, http://bugzilla.org

The rest of the paper is organized as follows. Sections 2 and 3 provide background on Jazz and on related projects. Sections 4 and 5 describe the course integration and evaluation, and Section 6 discusses lessons learned and future work.

2. BACKGROUND: WHAT IS JAZZ?

Jazz is a synchronous, collaborative development platform by IBM that integrates many different technologies into a single platform. The platform consists of both a client and a server. The server hosts many of the system configuration interfaces, the source code repository, and a web interface for generating reports and resolving work items. The Jazz client is based on Rational Team Concert, which is a derivative of Eclipse. Licenses for the Jazz development platform are available via the IBM Academic Initiative[4].

2.1 Jazz Features

Jazz Source Control is a traditional version control system with a focus on user tasks. The idea is for users to organize their tasks into multiple "change sets", which can be organized locally in the client, then committed to the repository separately. Team members can also track the local change sets of their teammates in real time. Team members can use this tracking capability to keep themselves updated on what their teammates are working on in synchronous situations. When team members commit new change sets to the repository, the other team members are notified immediately via a pop-up window in their client.

Jazz source control also provides other features that are difficult in similar tools. Temporarily reverting a change set is a simple click of the mouse and is performed at the local level (as opposed to branching and merging which is required in tools like Subversion). By allowing change sets to be temporarily reverted, users can switch between tasks easily while maintaining logical change sets.

The *Jazz Chat* view is a window in the Jazz client for instant messaging via typed text with teammates. The Chat view can handle multiple conversations and team conferences.

Jazz Work Items is the task and defect management feature in the development platform. Team members can open, discuss, attach information to, and resolve work items. Various kinds of information can be attached to a work item, such as a screenshot, a patch, a Chat conversation, or a change set. Opening a new work item is integrated into several different features, for example, right-clicking on a failing JUnit[5] test and creating a new work item automatically loads the relevant failure information into the new work item. Since work items can be assigned to different team members, one can keep track of his or her own work items using feeds. Work items can also be planned for completion at a particular date or iteration.

2.2 A Typical Sequence of Events

To demonstrate how all of the features of Jazz integrate together, consider the following flow of events. Suppose you are working on your project, and you accept a change set from one of your teammates into your local workspace. The new feature works, but now a JUnit test is failing. Initially, you think the code you are currently working on might be conflicting with your teammate's code, so you temporarily revert your current change set with a single click in the Pending Changes view. The test is still failing, so you create a new defect report by right-clicking on the failing test. The new work item is populated with all of the relevant failure information, so you simply assign the item to your teammate and save the new work item.

Immediately, your teammate gets a notification that a work item is assigned to him, and sends an instant message to you in the Chat view. After a short discussion, you decide that the test case needs updating, so your teammate fixes the test case, attaches the change set to the work item, attaches the instant message conversation to the work item, commits the new change set, and resolves the work item – all with a few clicks of the mouse. The resulting work item now contains all of the relevant artifacts relating to its resolution. An annotated screenshot of a work item is shown in Figure 1, with (a) failing test case information, (b) associated chat conversation, and (c) a link to the associated change set with the mouse hovered over it.

3. RELATED WORK

Although Jazz provides a unique integration of many different development features, other educators have reported similar successes with distributed development. We examine some of these projects here.

Reis et al. [9] present their project DrJava as a pedagogical integrated development environment for teaching introductory programming. Among their evaluation criteria of development environments is to have a simple, yet powerful interface with integrated features. Much of the success of DrJava is from having the features be integrated, which is also a key characteristic of Jazz that led to our success. DrJava is intended for first- and second-year students while our course was for third- and fourth-year students.

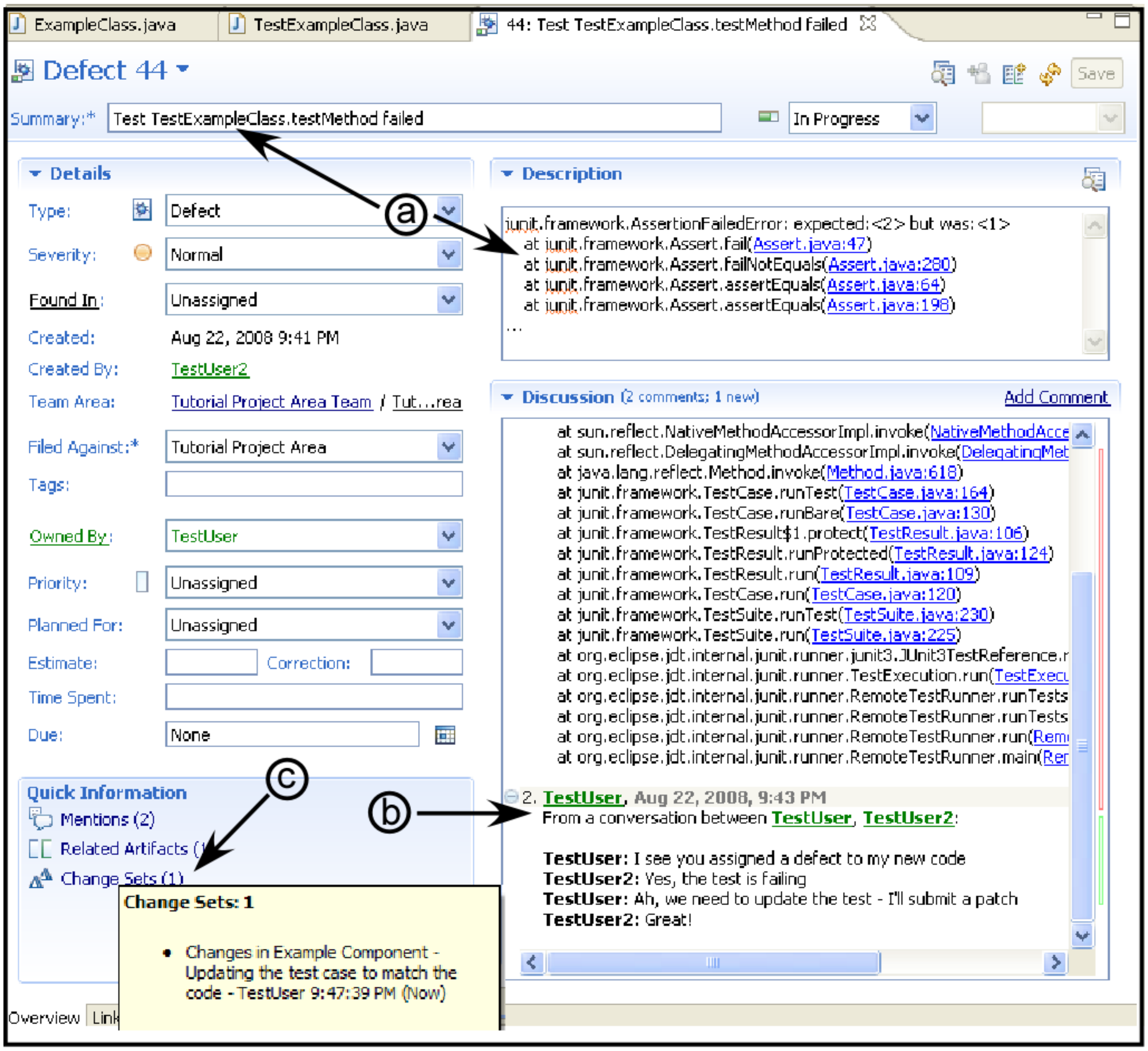
Reid and Wilson [8] present DrProject, a software project management system built specifically for undergraduate software engineering education. The main features of DrProject are version control, issue tracking, mailing lists, and a wiki. The goal of DrProject is to integrate many technologies into one web-based application. While the goals of DrProject are similar to that of Jazz, DrProject is primarily web-based. One of the advantages of Jazz is that most of the team features are integrated directly into the development environment.

Many educators, such as Clifton et al [1] and Hartness [5], have emphasized the use of version control in the classroom as a way to promote collaboration and to allow the instructors to monitor student activity. Our software engineering course is the first course in our curriculum that uses version control, and our experience has shown that students react very positively to using version control.

4. COURSE INTEGRATION

We used Jazz as an extra credit, optional development platform for the team project of our third- and fourth-year undergraduate software engineering course in the spring semester of 2008. Our usual environment was Eclipse with Subversion and Bugzilla.

[4]http://www.ibm.com/university/academicinitiative
[5] http://junit.org. JUnit is a unit testing framework for Java.



ExampleClass.java | TestExampleClass.java | 44: Test TestExampleClass.testMethod failed

Defect 44

Save

Summary:* Test TestExampleClass.testMethod failed

In Progress

Details

Type:	Defect
Severity:	Normal
Found In:	Unassigned
Created:	Aug 22, 2008 9:41 PM
Created By:	TestUser2
Team Area:	Tutorial Project Area Team / Tut…rea
Filed Against:*	Tutorial Project Area
Tags:	
Owned By:	TestUser
Priority:	Unassigned
Planned For:	Unassigned
Estimate:	Correction:
Time Spent:	
Due:	None

Description

```
junit.framework.AssertionFailedError: expected:<2> but was:<1>
  at junit.framework.Assert.fail(Assert.java:47)
  at junit.framework.Assert.failNotEquals(Assert.java:280)
  at junit.framework.Assert.assertEquals(Assert.java:64)
  at junit.framework.Assert.assertEquals(Assert.java:198)
...
```

Discussion (2 comments; 1 new) Add Comment

```
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAcce
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMet
at java.lang.reflect.Method.invoke(Method.java:618)
at junit.framework.TestCase.runTest(TestCase.java:164)
at junit.framework.TestCase.runBare(TestCase.java:130)
at junit.framework.TestResult$1.protect(TestResult.java:106)
at junit.framework.TestResult.runProtected(TestResult.java:124)
at junit.framework.TestResult.run(TestResult.java:109)
at junit.framework.TestCase.run(TestCase.java:120)
at junit.framework.TestSuite.runTest(TestSuite.java:230)
at junit.framework.TestSuite.run(TestSuite.java:225)
at org.eclipse.jdt.internal.junit.runner.junit3.JUnit3TestReference.r
at org.eclipse.jdt.internal.junit.runner.TestExecution.run(TestExecu
at org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.runTests
at org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.runTests
at org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.run(Rem
at org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.main(Rer
```

2. **TestUser, Aug 22, 2008, 9:43 PM**
From a conversation between **TestUser**, **TestUser2**:

TestUser: I see you assigned a defect to my new code
TestUser2: Yes, the test is failing
TestUser: Ah, we need to update the test - I'll submit a patch
TestUser2: Great!

Quick Information
Mentions (2)
Related Artifacts (1)
Change Sets (1)

Change Sets: 1

- Changes in Example Component - Updating the test case to match the code - TestUser 9:47:39 PM (Now)

Overview | Link

Figure 1: Work item from example flow of events in section 2.1.2 with (a) the failing test case information, (b) the chat conversation, and (c) the associated change set for the fix

The team project is a five week project at the end of the semester. As a required class for our Computer Science curriculum, software engineering is intended to prepare students for a career in professional software development. The course, taken by nearly 50 third- or fourth-year students, is comprised of two weekly 50-minute lecture sessions and one weekly two-hour lab session. The lab sessions provide students with hands-on experience through tutorials and other teaching activities. The first author of this paper was a lab instructor for the course.

Leading up to the team project, the course has four major assignments over the semester, each directly working with iTrust,[6] an open-source, role-based healthcare J2EE web application. Throughout the semester, all of the assignments are based on iTrust and consist of maintenance, automated testing, design and developing new features specified in requirements given by the teaching staff.

At the beginning of the semester, the students learned Eclipse, JUnit, Subversion, Bugzilla, and several other technologies to work on iTrust. A few weeks prior to the team project, however, we introduced Jazz to the students as an alternative to their current development environment.

The team project is the culminating assignment for the course and is meant to teach students about working together on a large project. The teaching staff provides a fully updated set of requirements with new features and changes to old features. The lab instructors place the students into teams of three or four members according to a personality test[7]. Over the five week period, students complete four iterations at the end of which they are asked to demonstrate their work to the lab instructor. Upon initial formation, the teams need to decide whether to use Jazz or Eclipse/Subversion/Bugzilla. Three groups out of a total

[6] http://agile.csc.ncsu.edu/iTrust/

[7] Prior studies [10] indicated Myers Briggs Sensors work well with Myers Briggs Intuitors.

of 11 groups, or 11 students out of a total 48, chose to use Jazz for the project.

Although we have a lab dedicated to software engineering, the majority of the students work on the team project from home. Historically, students use their team wiki, instant messaging, and email to stay synchronized.

5. EVALUATION

In this section, we summarize the student retrospective essays of Jazz, and examine differences in performance between Jazz and Eclipse groups.

5.1 Student Retrospectives

At the end of the semester, every student was required to submit a retrospective essay answering several questions about their experience with the course. The students who chose to use Jazz as their development platform were required to answer three additional questions:

Q1: Describe your overall experience with using Jazz in your team project.

Q2: Which features of Jazz did you use? Which features did you find easy to use? Hard to use?

Q3: Would you support using Jazz in future semesters of this course?

The *Jazz source control* was the most positively reviewed feature in the retrospectives. Nine out of the eleven students chose to highlight the source control as easy to use and/or helpful. Students particularly enjoyed being able to track the changes of their teammates in real time. Five out of the eleven students mentioned that the pop-up notification of changes in the repository was a helpful feature as they often worked remotely and synchronously (as stated in their retrospective). The groups with Eclipse/Subversion/Bugzilla did not have a notification feature, so they would not have known that their teammates were working at the same times. One student said the following in his retrospective:

> *The configuration management tool in Jazz was much better than subversion. Whenever an update was made by another team member a [pop-up] window would notify you. Comments were required for each submission and could be applied to resolve work items.*

The only aspect of the Jazz source control feature that was noted as confusing or difficult was handling code conflicts – an interface we did not cover in our tutorials and lab exercises. Three students found the interface confusing and would have liked more practice resolving source control conflicts before their project began.

The *Chat* feature was used heavily by two out of the three groups. Students reported that, though they would have used instant message clients anyway, they enjoyed having the chat window in their development environment.

All three groups reported using the *Jazz Work Items*, however, five of the students deemed the feature to be too complex to be useful. These reviews, however, are better than what we get regarding Bugzilla. Although both Jazz Work Items and Bugzilla have similar feature sets, students seldom use Bugzilla when they are not mandated to as they find it inconvenient. The group who relied heavily on Jazz Work Items, however, found the Work Items to be a valuable task management tool. One student reported the following in his retrospective:

> *With the work items feature […] everyone knew what needed to be done. When a team member elected to work on an item they could mark it as owned by them and in progress so that everyone else knew it was in work and by whom. This prevented duplication of effort and allowed each member to work on a unique item.*

For question Q3, students were asked if they would support using Jazz in future semesters of the course. All of the students reported that they would support using Jazz in future semesters. Students reported that using Jazz for the entire semester would help the learning curve. Also, students asked for more demonstrations, tutorials, and lab activities to assist in the initial learning. One student reported the following in his retrospective:

> *I think it is very feasible to use Jazz for the entire course and start with it from day one. […] I think Jazz is a very powerful tool and is not difficult to use.*

For Fall 2008, we are using Jazz for the entire semester, and we are writing additional tutorials[8] and lab activities for Jazz.

5.2 Analysis of Grades

In addition to examining student feedback on Jazz, we also evaluate overall student success in the team project. For this purpose, we examine the difference in grades between the Jazz and Eclipse teams.

The grading for the team project is designed to assess how well students worked in groups, implemented new features, met weekly deadlines, and improved the overall reliability of the product and its test suite. The team project was graded according to the following breakdown: initial project planning (20%), three weekly iterations (20%), two peer reviews (10%), unit and system testing (10%), overall product reliability (27%), static analysis (3%), and the retrospective (10%). While most of the grades are based on the team's collective performance, adjustments are made to the individual grades when we find evidence of lacking participation. Extra credit options are available but are not included in the analysis.

Table 1 shows that the average of the individual grades by development environment used. The resulting difference is statistically significant (Student t-test, unequal variances, $p<0.01$), indicating that students who used Jazz had a higher team project grade, on average.

Table 1: Average project grade for Jazz and Eclipse groups

Choice of IDE (#students, #groups)	Average Project Grade	Standard Error
Jazz (11, 3)	96.0%	0.74%
Eclipse (37, 8)	90.1%	0.87%

Lastly, we noticed that productivity increased with weaker students who used Jazz. For students whose grade was lower than 80% prior to the team project, the students who used Jazz

[8] http://agile.csc.ncsu.edu/SEMaterials/tutorials/

had a higher average grade project grade than the Eclipse students (Student t-test, unequal variances, $p<0.01$).

One possible confounding factor is that the stronger students may be more likely to choose an unfamiliar tool such as Jazz, meaning that the higher average grade is indicative of the students, not the tool. The difference in student grades prior to the team project, however, was not statistically significant between the Jazz and Eclipse groups (Student t-test, $p>0.1$ for both equal and unequal variances).

Note that this kind of analysis is does not show any indication of causation. Since there may be more confounding factors, we cannot definitively say that Jazz improved the student grades; we can only show that the grades in the Jazz groups were higher.

6. LESSONS LEARNED, FUTURE WORK

Among the more poorly reviewed characteristics of Jazz was learning the terminology. For example, a version control "commit" is called "deliver" in Jazz. In many retrospectives (and in person), students stated that they would have liked to have more demonstrations and tutorials to support their initial learning. We believe that by committing to a whole semester and by writing more tutorials using Jazz, students' initial learning will be improved.

Using Jazz for a whole semester will also give students time to explore more of Jazz's features. Students reported that they would have liked to use many of Jazz's other features such as the build system, code coverage, and iteration planning. We plan on using all of these features in future semesters. With the platform being integrated, we can teach more technologies without being distracted from our learning objectives. Covering more technologies will help better prepare our students professional development, a field which constantly requires learning new tools and technologies.

Although we found that Jazz is a great tool for synchronous development, it did not have support for distributed pair programming (i.e. two people working on the same file at the same time). We are currently evolving and evaluating a distributed pair programming tool for Jazz based on the Eclipse plug-in Sangam[9], called Jazz Sangam [3]. With Sangam, teammates can view each other's screens remotely so that two developers work on a single resource at once.

7. CONCLUSION

Working remotely is becoming the norm for both professionals and students alike. We found that using the synchronous, collaborative development platform Jazz in a software engineering course improved the overall experience and success of our students. All of the eleven students who used Jazz as their development platform supported using Jazz in future semesters of the course. Using Jazz in the context of a team project, students used many of the synchronous development features of the system to work as a team. By teaching with an integrated platform, we can cover more technologies that are similar, if not the same as, what students will encounter in their careers.

[9] http://sangam.sourceforge.net/

8. ACKNOWLEDGMENTS

Funding for this work was provided by an IBM Jazz Innovation Award. We thank Lauren Hayward and Tao Xie for technical support with Jazz. We also thank the Realsearch reading group and Jeremy Frens for their valuable comments on this paper.

9. REFERENCES

[1] Clifton, C., Kaczmarczyk, L. C., and Mrozek, M., "Subverting the Fundamentals Sequence: Using Version Control to Enhance Course Management," in *38th SIGCSE Technical Symposium on Computer Science Education*, Covington, Kentucky, USA, 2007, pp. 86-90.

[2] Damian, D., Hadwin, A., and Al-Ani, B., "Instructional design and assessment strategies for teaching global software development: a framework," in *28th International Conference on Software Engineering*, Shanghai, China, 2006, pp. 685-690.

[3] Devide, J. V. S., Meneely, A., Ho, C.-W., Williams, L., and Devetsikiotis, M., "Jazz Sangam: A Real-Time Tool for Distributed Pair Programming on a Team Development Platform," in *Workshop on Infrastructure for Research in Collaborative Software Engineering*, Atlanta, GA, 2008, p. to appear.

[4] Fryer, K. and Gothe, M., "Global Software Development and Delivery: Trends and Challenges," IBM Developerworks,http://www.ibm.com/developerworks/rational/library/edge/08/jan08/fryer_gothe/index.html, 2008, accessed on August 23, 2008.

[5] Hartness, K. T. N., "Eclipse and CVS for group projects," *Journal of Computing Sciences in Small Colleges*, vol. 21, no.4, pp. 217-222, 2006.

[6] Hawthorne, M. J. and Perry, D. E., "Software engineering education in the era of outsourcing, distributed development, and open source software: challenges and opportunities," in *27th International Conference on Software Engineering*, St. Louis, MO, USA, 2005, pp. 643-644.

[7] Herbsleb, J. D. and Mockus, A., "An Empirical Study of Speed and Communication in Globally Distributed Software Development," *IEEE Transactions on Software Engineering*, vol. 29, no.6, pp. 481-494, 2003.

[8] Reid, K. L. and Wilson, G. V., "DrProject: a Software Project Management Portal to Meet Educational Needs," in *SIGCSE*, Covington, KY, 2007, pp. 317-321.

[9] Reis, C. and Cartwright, R., "Taming a Professional IDE for the Classroom," in *35th SIGCSE Technical Symposium on Computer Science Education*, Norfolk, Virginia, 2004.

[10] Williams, L., Layman, L., Osborne, J., and Katira, N., "Examining the Compatibility of Student Pair Programmers," in *Agile 2006*, Minneapolis, MN, 2006, pp. 411-420.