A PLATFORM FOR TEACHING DISTRIBUTED SOFTWARE ENGINEERING

Philipp Bouillon, Jens Krinke [1])

Abstract

Many problems in distributed software engineering (DSE) arise, because the participants of a team are not trained in DSE. We present an integrated development environment which supports collaborative working. To introduce such a system in the teaching of software engineering provides for a higher educational standard and for a better awareness of students for the problems involved in distributed software development. We present the necessary tools to accomplish the development of such an IDE based on Eclipse and discuss the problems involved.

1. Introduction

Programmers do not learn how to program at school or at a university. They are often self-educated individuals who not so rarely have difficulties integrating into existing teams or forming new ones. The problem gets even worse when it comes to working in a distributed environment. Not only will the programmers have to deal with a variety of people working on the same project, but they will also have to deal with the technical issues involved here. Some of these issues are asynchronous communication due to different time zones, a lack of tools for collaborative distance UML design, or the need to use a version control system in a defined and strict way. The inability of the programmers to deal with those issues is a reason why many open source projects fail.

Many of those problems can be made easier, if not completely eliminated, if the programmers already know how to deal with them. So, in teaching software development it becomes an increasingly important task to teach *distance collaborative working*. Our goal is to define an integrated development environment (IDE) which provides the needed tools to make remote working in a team not only easier but also more fun, thereby increasing the understanding of problems in the students' minds and thus, eventually, increasing the productivity of graduated programmers. If students are not aware of the problems in DSE, how will they ever be able to solve them?

As a base for our development tools we use Eclipse [8], as it is a highly extendable and open IDE especially useful for Java programming. The fact that the IDE is open is relevant for plug-ins supporting collaboration, communication, and coordination. However, the fundamentals discussed here hold for any other integrated development environment as well.

The remainder of this discussion is organized as follows: Section 2 describes the scenario which applies for teaching software development to students at the *FernUniversität in Hagen*; Section 3 analyzes the tools we are planning to integrate into our development environment, and Section 4 concludes the discussion with an overview of work which still has to be done.

[1] FernUniversität in Hagen, 58084 Hagen, Germany

2. Background

At the *FernUniversität in Hagen*, the first distance teaching university in Germany, software engineering is taught paying special attention to problems arising from the *distribution* of our students. At present, the students visit the university in Hagen to form their study groups, then they return to their respective homes and discuss their work via a system called CURE (Collaborative Universal Remote Education environment [5])—simply spoken a combination of wiki and chat, including a mail system, a simple calendar and a newsgroup mechanism. It is possible to perform a requirements analysis of any given programming assignment using only CURE: Students must discuss various aspects of the software and they will eventually discover that they all have different understandings of the vocabulary involved in the project. CURE supports both, synchronous and asynchronous communication, and as soon as the discussions are over, the result must be written down to precisely document the direction of the project. CURE allows for the result to be documented in its wiki which provides a very good alternative to LaTeX or Word. In short: CURE is *the* tool for performing a distributed requirements analysis.

For the design phase, students at the FernUniversität are required to use UML. At the moment, no possibility exists to use UML tools in a collaborative way, so once again, the students must visit Hagen in order to discuss and design their project. At the university, they find a variety of tools to draw UML diagrams.

During the implementation phase, all students work at home, but as yet, no means of sharing tasks have been used at the university (for example, if the students want to use *Bugzilla* or some other tracking tool, they have to set up a server on their own).

Software engineering at the FernUniversität takes DSE to the edge—usually, at any given place and any given time, there is not more than a single student working on the project. In this situation, the idea arose to completely *virtualize* the teaching of software engineering. Therefore, some server-based services must be provided by the university and a client platform must be delivered to the students—preferably at no cost for them.

3. Required Tools

This section gives an overview of the tools that are necessary to support a fully virtual education of software development processes. As a base tool, the Eclipse IDE is used, because it is highly extendable and open source.

3.1. Phase 1: The Teacher Creates the Project

When creating a project, a supervisor must be able to enter a project description and mark the milestones at which certain parts of the project have to be completed. For example, the supervisor can enter the due dates of the *phase documents* into a team-calendar which is shared by each team member, allowing for the addition of team-internal dates.

The optimal solution for this kind of (simple) calendar would be to integrate Eclipse with MS Outlook or Lotus Notes since these are the applications used by most people to organize their meetings. For our educational purpose, several open-source alternatives are considered but none of them could possibly compete with Outlook or Notes.

Plug-ins already available for Eclipse: None.

3.2. Phase 2: Requirements Analysis

To allow for a vivid discussion among team members, both, synchronous and asynchronous communication must be provided. Furthermore, the final requirements document should be produced in the same environment without the need to constantly switch between applications. Therefore, a chat and e-mail functionality, as well as wiki-pages where team members can jot down their ideas would be ideal.

Most of this has already been implemented in CURE where students can discuss the requirements and can use the wiki functionality to produce their resulting document. So, an integration of CURE with Eclipse is currently underway. Students may use it alone or together with a LaTeX-Plug-in (like LatEc [1]), to provide a more familiar way of producing documents. As soon as the integration is complete, the requirements analysis can be done completely in Eclipse. In the future, the development team of CURE plans to release CURE for companies as well, so it might just be an ideal solution for team-internal communication.

Plug-ins already available for Eclipse: LatEc.

3.3. Phase 3: Design

A variety of UML design tools exist. However, none of the available open source tools allow for *collaborative* design. Very useful would be a graphical *diff*-view that directly shows the user, which changes were made (i.e. by highlighting modified, added, or removed elements in the diagram).

Furthermore, UML design can be accompanied by a chat window to support synchronous communication during the work. A number of plug-ins for Eclipse already exist to support that, and once the integration with CURE is complete, a unified look-and-feel will be provided for the chat, too.

Plug-ins already available for Eclipse: Various UML-tools: Best suited for our needs are Together [3] and MagicDraw [6]. Chat is supported by an IM-Plug-in [4], by PepeMax [7] which supports Jabber, and by the JAZZ Project [13].

3.4. Phase 4: Coding and Module Testing

Eclipse is already excellently equipped for coding purposes in Java. Other languages are not so well supported, yet, but this is hopefully going to change in the future. For testing (and one way to specify units), JUnit is shipped with Eclipse and our students are required to test all of their units. Independent of the programming language, a team interface is integrated with Eclipse which allows for CVS communication, so this collaborative issue is already solved elegantly in Eclipse, although as yet, it does not provide the possibility to perform a graphical diff of UML diagrams. To improve this phase, external tools like JML (a specification language for java) will be provided. Furthermore, the GILD project [12] will be used to allow the students to assign various tasks to lines of code. Not only bugs can be marked in this way, but also questions from one developer to all other team members can *persistently* be asked and answered. GILD also allows for the instructors to annotate code parts and because of the persistent annotations, the tool also helps in grading as it is now possible to determine to which extent a team member contributes to the project.

Plug-ins already available for Eclipse: CVS, JDT, JUnit, CDT [2], GILD [12].

3.5. Phase 5: Bug Management

The key tool for managing bugs is Bugzilla. It will be provided by a server located at the FernUniversität and thus it can be accessed by any team member currently taking the software engineering course. Together with the calendar and task management functionality described in Section 3.1., the management of bugs is made easier for the members of a team. To provide access to Bugzilla from within Eclipse is difficult due to the problems of getting the specifications of Bugzilla; but other projects, like Insectivore [9] make up great alternatives.

Plug-ins already available for Eclipse: Insectivore.

3.6. So, What's the Benefit?

During a DSE course at the FernUniversität in Hagen, the students not only learn how to develop a working software system, they are also introduced to the tools just presented here and everybody will gain a profound understanding of the application of the tools as well as a deep knowledge of the theoretical foundations upon which they are built.

Based on earlier experience, we are of the opinion that students need strong support in the following areas:

1. Most communication and collaboration is asynchronous. Therefore, the students need strong support in asynchronous means. This does not only mean e-mail, news, wiki and so on, but moreover means strong support for *traceability*—what has changed when and why?
2. Experiments with agile methods showed that students tend to have problems in flexible environments, because they delay work to the last minute. Better results are gained with an explicit process and strict deadlines and milestones.

We plan to use *trackers* to address these issues; not only tracking bugs (with Bugzilla or Insectivore), but also tracking other tasks and problems. For that reason, we will integrate GForge [10], an Open Source collaborative software development tool, or CodeBeamer [11], a commercial solution.

One reason for the hesitant use of synchronous communication tools among students (aside from the fact that they work at different times), *may* be that they have to switch from their programming tool to the chat tool in order to send a message to a team member. It remains to be seen if the integration of existing chat functionalities with Eclipse will alter this behavior. JAZZ [13] will be a part of our IDE so that the students will have several possibilities to communicate—both synchronous and asynchronous.

4. Conclusion

To provide an integrated development environment for collaborative learning of software engineering is difficult. Yet, with the advent of Eclipse and the large variety of plug-ins, it becomes a feasible task to provide an environment which comes with the ideal tools to learn *the difficult part* of software engineering—to be part of a distributed team.

It is thus to be expected that team members who are trained in DSE will be far more productive in industry-standard DSE projects, thereby leading those projects to success. Since the teaching of software engineering at the FernUniversität in Hagen is always *distributed software engineering*, this project will eventually establish standards and processes for DSE education which will then be transferred to traditional universities and later to the industry for in-house training.

It remains a lot to be done in order to achieve the goal of a completely integrated development environment, but it cannot be doubted that the industry will benefit from the end-result: Students who already know of the difficulties involved with remote software development and students who are aware of some of the solutions and willing to provide their share of experiences and ideas.

5. References

[1] BORELLA, J., LaTeC—LaTeX Eclipse plug-in. <http://www.itu.dk/people/jborella/> [Accessed 12 Aug 2004].

[2] CDT TEAM, CDT—C/C++ Development Tools. <http://www.eclipse.org/cdt/> [Accessed 12 Aug 2004].

[3] BORLAND SOFTWARE CORPORATION, Together Edition for Eclipse. <http://www.borland.com/together/eclipse/index.html> [Accessed 12 Aug 2004].

[4] Eimp—Eclipse Instant Messenger plug-in. <http://eimp.sourceforge.net/d/> [Accessed 12 Aug 2004].

[5] HAAKE, J., SCHÜMMER, T., HAAKE, A., BOURIMI, M., LANDGRAF, B., Supporting flexible collaborative distance learning in the CURE platform, Proceedings of the Hawaii International Conference On System Sciences (HICSS-37), 2004.

[6] NO MAGIC, INC., Magic Draw UML. <http://www.nomagic.com/> [Accessed 12 Aug 2004].

[7] PepeMax—a Jabber Plug-In for Eclipse. <http://pepemax.jabberstudio.org/> [Accessed 12 Aug 2004].

[8] OTI LABS, Eclipse. <http://www.eclipse.org/> [Accessed 12 Aug 2004].

[9] Insectivore. <http://www.zclipse.org/projects/insectivore/> [Accessed 12 Aug 2004].

[10] GForge CDE. <http://www.gforge.org> [Accessed 12 Aug 2004].

[11] INTLAND, CodeBeamer. <http://www.intland.com/products/codebeamer.htm> [Accessed 12 Aug 2004].

[12] STOREY, M.-A.; SANSEVERINO, M; GERMAN, D.; DAMIAN, D.; DAMIAN, A.; MICHAUD, J.; MURRAY, A.; LINTERN, R.; CHISAN, J.: Adopting GILD: An Integrated Learning and Development Environment for Programming, International Conference on Software Engineering (ICSE), Portland, USA, May 2003.

[13] CHENG, L.-T..; HUPFER, S.; ROSS, S.; PATTERSON, J.: Jazzing up Eclipse with Collaborative Tools, in Proceedings of the 2003 OOPSLA Workshop on Eclipse Technology eXchange, Anaheim, USA, October 2003.