Teaching Distributed Collaborative Development Techniques in a Software Engineering Class Setting

Gary D. Thompson
SFSU and SUN Microsystems
4150 Network Circle
Santa Clara CA 95054 USA
gary.thompson@sun.com

Dragutin Petkovic
San Francisco State Univ., CS Dept.
1600 Holloway Ave, TH 906
San Francisco CA 94132 USA
dpetkovic@cs.sfsu.edu

Shihong Huang
Dept. of Computer Science & Engineering
Florida Atlantic University, USA
shihong@cse.fau.edu

Abstract

In this paper we present a synopsys of a Global Software (SW) Engineering class based on several years of experience of teaching jointly at San Francisco State University (SFSU), the University of Applied Sciences, Fulda University, Germany, and recently with Florida Atlantic University (FAU). This class uses numerous Free and Open Source Software tools and teaches FOSS development techniques. We define global SE practices as those used when the team members are distributed in terms of location and time, and hence do not meet regularly in person. Global SE practices have become a significant mode of SW development, especially in the development of large open source software projects. Over the course of teaching this class from 2004 till present we have published several papers [12, 13, 18, 19] describing this class and the development of techniques for measuring levels of participation within a small software development group. This paper is an adaptation of those previous papers specifically designed to draw attention to FOSS tools, philosophies and development techniques.

Introduction

SW engineering (SE) practices today increasingly involve teams that are geographically distributed (whether in the same company or not), have team members with different cultural backgrounds, and involve complex communication and development practices over different time zones (we call them "global" teams). Training, educating and assessing efficacy of SE education and practices is complex, and only gets more complicated with introduction of these geographically distributed global teams. Recent studies such as [5, 6, 7, 9, 11, 16] indicate that there are distinct advantages to be gained from creating globally distributed development teams, as well as challenges, many of these cultural, to be overcome. The challenge also remains as to how to effectively prepare and teach students and employees for this environment [10, 12, 13, 14, 15, 17].

At SFSU we have been teaching SE class for the last four years in conjunction with Fulda University in Germany, and this year with both Fulda University and for the past year with Florida Atlantic University (FAU). As part of our teaching we use the class experience and measurements to: a) understand differences between global and local SE practices from the developer and team lead perspective, b) to develop better teaching methods for practical and global SE, see [12, 13, 17], c) to develop automated and semi-automated tools for assessment and monitoring of global SE practices. Our class teaching methods combine classical teaching of SE methods and practices (using guidance and textbooks from [1, 2, 3, 4, 7]) synchronized with an intense final group project, where student group members act as software developers (usually 4-6 students in a group and 8-12 groups per class) and build a SW application using FOSS development techniques.
Local student groups are comprised only of graduate and undergraduate students from SFSU, or Fulda or FAU, while each global group have students from SFSU-Fulda, and SFAU-FAU. The students from SFSU never meet in person with the students from Fulda or FAU. Team interaction among students and with instructors at each school is required, strongly encouraged, and is monitored. Each student team has instructor-appointed team lead. Each student group develops a complete Internet application following all major steps of full SW development lifecycle, from requirement solicitations to final delivery, in five formal milestones: (1) high level requirements; 2) more detailed requirements and specs; 3) prototype; 4) beta delivery; and 5) final delivery and demo. These milestones are documented, revised upon instructors' feedback and submitted with final "SW product" for grading. FOSS development techniques such as peer support through public mailing lists and discussion forums and documentation of specifications and the discussion that lead to the acceptance of those specifications on publicly viewable websites and in publicly accessible mailing lists and release of a publicly hosted source software project as a final project.
During the course of teaching this class we have refined our original set of measurements used to compare local and global SE engineering practices and also to assess student learning outcomes related to SE and teamwork. Based on four years of class experience we have now identified five factors that we suggest can be used to analyze global SE practices and compare them with traditional SE practices. These five factors are as follows: 1) *Quality of Final Deliverables* (product and milestone documentation quality, but also the adherence by students to SE process)*; 2) Progress Impediment Factors* such as physical distance, cultural differences and time zone differences between team members; 3) *Effort Expended* e.g. the amount of work including SW development, documentation development, meeting times, e-mail, VOIP usage etc. measured in time; 4) *Collaborative Activity* e.g. involvement by all team members in terms of on-line communication and development of deliverables (e.g. number of postings); and 5) *Teamwork problems.* Note that these factors specifically focus on SE practices and measurements and *not* on assessment of programming proficiency. We believe that these findings are also relevant to real-world global and local SE practices. In this paper we will outline the goals and objectives of our class, describe the teaching methods and FOSS tools we use, the FOSS development techniques we teach and describe the participation metrics collection software we have developed.

Goals and Objectives of the Distributed Collaboration Software Engineering Class

In Fall 2004 SFSU and University of Fulda started a new Distributed collaborative Software Engineering class to address the needs of modern CS students with a novel approach to SE education. Since that time the SE class from SFSU has worked in conjunction with

the Software Engineering class from the University of Applied Sciences, Fulda, Germany (http://www2.fh-fulda.de/isu/2005/fh_en.htm) to add an element of globalization to the educational experience. In Fall of 2008 Florida Atlantic University, FAU (http://www.cse.fau.edu/) joined into this collaboration with their Software Engineering class. In the Computer Science Departments of all three universities students pursue BS and MS degrees in Computer Science. Most students aspire to work in the high tech industry. A significant proportion of BS students continue toward a MS degree and some MS students (5-10%) chose to pursue a Ph. D. at other Universities. All students engaged in this class from University of Fulda are MS candidates. These universities also maintain a large population of foreign students. Entry surveys show that although many students have significant work experience, most students enrolling in the SE class do not have significant teamwork or project experience, nor do they have experience in the SE processes.

The high level goals of this SE class are:

A. To provide practical and competitive SE education for SW developers, tech leads and managers with focus on organization, communication, process, teamwork, and peer support including globalization issues and Free and Open Source Software tools and development techniques.
B. To analyze, understand and improve SE education (including the teaching of global SE).

Note that other components of the SE education, such as technical requirements, as recommended by [1], are provided in other classes and that students are expected to have considerable programming knowledge before joining this SE class.

The learning outcomes we expected from students successfully completing our SE class are consistent with [1, 9] and included:

2.Knowledge of basic software engineering and FOSS development methods and practices, and their appropriate application
3.Knowledge of basic components and tools, including FOSS tools, of full SW development lifecycle, from initial specifications to final documentation
4.Knowledge of design and build practices for easy to use, maintainable SW developed using modern multi-tier architectures
5.Knowledge of basic SW dependability metrics, quality metrics, and basic architectural models
6.Ability to constantly iterate and re-prioritize goals based on user needs, budget, schedule and resources
7.Development of significant teamwork and project based experience, as close as possible to real life
8.Exposure to global SE methods and practices.

Teaching Methods

Our SE class combines and synchronizes traditional classroom teaching with a significant class project developed by groups of students, including intensive and constant interactions between instructors and (local and global) student groups.

At high level, to address goal A) our course covers the main SE topics in a typical classroom teaching mode, such as:

[1]Overview of basic SE and FOSS development methodologies with emphasis on Iterative and Incremental Development and User Centered Design
[2]Usability and UI Design Principles and Practices
[3]Basic Components of SW Engineering Process: Requirements and Specifications; Iterative Design, Rapid Prototyping, Mockups; Software Design; Coding and Documentation Techniques (high level only);
[4]Open Source SW Development and Management
[5]Software Configuration Management, Delivery, Installation, and Documentation
[6]Software Metrics, Software QA and Testing
[7]Software Architecture and Middleware (high level only)
[8]Project Management issues:
[9]Teamwork and Communication
[10]Issues related to global SW engineering
[11]Real life examples and cases from instructor and students.

The books [3,4] are primary classroom material. For issues related to global SE we use [7] and various papers.

In parallel and synchronized with the teaching of the above topics students are formed into groups of 4-6 members. They are required to build a complete WWW application for a final class project by applying the SE and FOSS theory they learned in the class, while simulating a small SW company. We form "local" groups with students from SFSU, Fulda and FAU respectively (usually around 10 groups), as well as about 3-5 "global groups" consisting of students from two universities, SFSU and Fulda and SFSU and FAU (3 from each university). Global groups have the same tasks and milestones as local groups, and are offered a bit more coaching and help from instructors, as well as additional tools for distributed collaboration. Instructors play the role of senior management, and give user feedback. This group activity and intensive interaction with instructors simulating small SW companies, with development milestones synchronized with classroom teaching, is, in our opinion, critical for providing modern and practical SE education.

To address goal B), we have students fill out timely (weekly) questionnaires and provide written feedback, which we analyze and use to improve the class itself in an ongoing manner.

In order to encourage interaction and learning of SE and FOSS development methods, grading of this final project is equally based on the following criteria: a) adherence to SE and FOSS development methods b) student interaction, and c) quality of final delivery (documentation, ease of use, functionality, architecture etc.). In order to make teaching as effective as possible, and the simulation of real-life SW lifecycle as realistic as possible, the following methodology is used: a) instructors play the roles of "management (coaching) staff" (CEO, VIP Engineering, VIP Marketing, CTO, customer representative etc.) and customer advocates. Students are the developers. b) student team members are selected by the "management staff" i.e. instructors; c) management staff provid only the initial high level project concept, and the students themselves develop real requirements and specs following the appropriate methodologies; d) management staff impose a strict delivery timetable in the form of five well defined milestones, as follows:

- High-level requirements and specifications
- Detailed requirements and specifications (including functional, non-functional specs, use cases, priorities, mockups of UI, high level architecture, data and content dictionaries etc.)
- First prototype (with formal one hour walk through meeting outside of class);
- Beta launch with formal usability evaluation performed by instructors;
- Final delivery and demo in a simulated "trade show" presentation

Classroom teaching is synchronized in order to prepare students for each milestone, and at least 45 min. per week is used for group exercises where student groups meet while instructors perform "management by walking around".

The instructors collaborate before the class to determine a suitable WWW-based product concept for the final class project and to prepare the development environment. The projects completed over the four-year period include WWW applications for a video rental

store and online career service, an online real estate sales service and a semi-automated tool for collecting participation metrics within software development groups. A one page high-level description of the project (intentionally vague, and not necessarily complete) is the only document provided to the groups as the development assignment. Using the above five predefined milestones, and interacting with "management staff and customers" (i.e. instructors) the students are required to design, develop and demonstrate the full product. Students develop and document each milestone after doing their own designs and evaluations, while interacting with instructors. Each milestone is formally reviewed by instructors (but not graded at that time) and feedback is provided. Each milestone is then revised by students before moving forward, thus promoting iterative development and teaching students how to deal with constructive feedback. Documentation from all milestones is combined to form a documentation package for final class project delivery and grading. During the final milestone, each group demonstrates their project to the entire class. This is graded for overall quality, functionality, ease of use, implementation and architecture. This project is then released as an open source software project.

A very important component of the class is the instructors' interaction with student groups. It is made clear to students that iteration and interactions is encouraged and not graded, so that they feel comfortable asking questions, getting clarifications and challenging mandates. This interaction includes both "user feedback" (on functionality, UI, ease of use etc.) as well as technical and organizational support.

It was important to create student groups with a balanced set of skills. The SFSU class involves graduate and undergraduate students. In order to accomplish this balance, a survey on background and experience of each student is done at the beginning of the class. Determining criteria included a self-assessment of experience in:

- General programming
- Web based development including browser GUI
- Working in groups including distributed groups
- Project management
- The desire to work in a global group

Management staff then select group members, including global group members, to best balance the skill level of all groups. Starting Fall 2005, based on student feedback and previous experience, a student group lead is selected by management staff in order to facilitate communication between each group and the instructors. Beginning Fall 2008 a Milestone 0 was added that included downloading and installing all software development tools, following tutorials and learning trails for each of the development tools and then posting the results of those tutorials in a classwide Subversion (SVN) repository.

Development and Collaborative Tools Used

Tools provided to effectively complete the project are of two types: *collaboration* tools and *development* tools. For collaboration tools we chose java.net. Java.net is a free collaborative software development hosting site (http://www.java.net/). It provides various professional collaborative development tools including:

- Archived email lists
- Archived discussion forums
- SVN repository
- File sharing
- Issue tracking
- Customizable user interface

A public parent project (i.e. http://ppm.dev.java.net) is created to serve as the corporate intranet. This collaborative aspect was useful for the local development teams as well as the global teams. Mailing lists in this project are used for instructors to communicate with each other to communicate with all student involved in the project from all universities, with all Group Leaders and for all students to communicate with each other and with instructors. Discussion forums are provided on various pertinent topics to allow students to post and answer questions about the development tools, the project ,etc in order to develop peer support techniques.

Each student group is then assigned a project on java.net where they develop implement their solution to the project assigned. They use all the same collaborative tools to create an open source application. This project space houses all the documentation developed over the course of the term and with each milestone and the SVN repository houses the source code developed for the project. At the completion of the academic term this project space on java.net is then available as an "online portfolio" of development work that students can use during a job search to display and demonstrate to potential employers the type and level of development and collaborative work in which they have participated.

Additional collaboration tools provided to all groups include a VoIP telephony and chat service (Skype http://www.skype.com/) and a Virtual Network Connection (TightVNC, http://www.tightvnc.com/) server and client to allow shared computer desktop. For presentations of global groups, a web browser is run on a server in Fulda, and students from SFSU connect to this server with their browser, and use phone or VoIP for audio.

The development tool set typically includes a choice of two programming languages, Java EE or PHP, as well as a relational database (mySQL, http://www.mysql.com/), an application server when required (GlassFish https://glassfish.dev.java.net/), an IDE (NetBeans, http://www.netbeans.org/) and an SVN client (SVN, http://subversion.tigris.org/) which integrates with NetBeans for repository control. An application hosting service (Sunfire T2000 server, http://www.sun.com/servers/coolthreads/t2000/) loaded with required software is also provided as part of the SFSU CS infrastructure. Local student groups typically have a choice of selecting PHP or Java (making this choice was actually part of their class assignment), while global groups are typically required to use Java. In Fall 2008 Java EE as a development language was not offered as an option.

Instructional Plans Implemented Fall 2008

The measurements and analysis of SE practices we have developed in our classes over several years of experience serve as a good starting point for assessment, analysis and comparison of global and local (traditional) SE. It is very desirable to automate and facilitate these measurements as much as possible, make them integrated with actual tools used in SW development and collaboration, as well to make them easily accessible. Such automation would provide a useful tool for managers to monitor group activities, researchers to analyze and compare global and local SW engineering and for educators to assess student learning progress. Some of the measurements we proposed can be automated (such as activities like posting, commits etc.), and some require human intervention and input, such as self study questionnaires, instructor abbreviations etc. As a class project this year, we have chosen a project called Project Participation Metrics (https://ppm.dev.java.net/). This application is

intended to automate and enhance the metrics used to assess collaborative activity and teamwork problems.
The collaborative activity statistics are currently provided by the tools archives provided by java.net. However, it is required to manually visit each mailing list or discussion forum (SCM commits are announced through a dedicated mailing list in each project) for viewing the archives and extracting the information. By this process it is possible to see the total number of posts per tool and, by drilling down through the archives, to determine the date, author and contents of the post. In the case of the SCM repository contents covers the names of the files that are modified and committed to the repository. This makes it possible to get a sense of the overall activity of each team, but difficult to assess the contributions of individual members. Teamwork issues are recorded and maintained by a log kept by instructors. This tends to make the current method of collection of these measurements difficult, tedious and unscalable.
The Project Participation Metrics application, which each group will be tasked with implementing will use the basic data gathering from current java.net and will develop of a browser based service that performs two main tasks:
1. On a regular basis the application will visit the collaborative platform, locate the projects of interest and then drill down into those archives to extract pertinent information provided by each collaborative tool there. On java.net this information is currently published by an RSS feed associated with each tool. This information will then be deposited into a database for use by other aspects of the application. This portion of the application will require a login by a project owner to search private projects.
2. Provide a browser based user interface that allows an instructor to effectively use the application. This user interface will have 4 aspects:

- A configuration form that will allow the instructor to set up the project names and URL's that will be searched by the application.
- A browser based input interface that will allow instructors to log and document teamwork problems or other issues and then record them into the data base for later use.
- A browser based user interface that will allow instructors to develop queries into the data base to help analyze collaborative behavior of an individual or a set of students. Results of the query should be displayed in either tabular or graphical format and a CSV file will be provided as an optional output.

Students are provided with a demonstration file that will reach out to java.net and collect the required information from the RSS feeds. This demo application will need to be refactored to work within student own applications and expanded to provide the additional functionality.
Each of the implementations of this project will be made an open source project hosted within Education community on java.net where they will be available for download and use by other instructors.

CONCLUSIONS

In this paper we presented objectives, teaching methods and tools used to teach our global software (SW) engineering class jointly taught for the last four years between San Francisco State University (SFSU) and the University of Applied Sciences, Fulda University, Germany and recently with Florida Atlantic University (FAU). We also outlined our most recent effort to automate some of those participatory measurement data points and make them easily accessible.
One can conclude that nowadays: a) all SW teams, local or global, are diverse and therefore culture issues have to be addressed in both of those modes of operation; b) local and global teams will most likely work significantly without physically meeting together, and c) many developers will work in the FOSS development sector. Thus it is very important to take this into account and provide adequate communication infrastructure , FOSS tools and development techniques and training on teamwork behavior and polices, whether the teams are local or global.
Finally, based upon the work done this past term we plan to develop an automated assessment tool integrated with collaborative tools. This tool will be useful for instructors teaching SE for monitoring and analysis of collaborative SE practices.

Additional Authors

Rainer Todtenhoefer
Univ. of Applied Science, Fulda
Marquardstr. 35
D - 36039 Fulda, Germany
Rainer.Todtenhoefer@Informatik.fh-fulda.de

References

[1] "Software Engineering 2004": Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering, The joint Task Force on computing Curricula, IEEE Computer Society, Association for Computing Machinery, August 2004 http://sites.computer.org/ccse/SE2004Volume.pdf
[2] Charette, R. N., “Why Software Fails”, IEEE Spectrum, September 2005, pp. 42
[3] Somerville, Ian, "Software Engineering", 6th edition, Addison Wesley, 2000 (7th edition also applies)
[4] Torres R. J., “Practitioner’s Handbook for User Interface Design and Development”, Prentice Hall PTR, 2002 [Carmel 1999] E. Carmel : “Global Software teams” , Prentice Hall, 1999
[5]Krishna, S, Sahay,, S, Walsham G, “Managing Cross-Cultural Issues in Global Software Outsourcing”, Comm. Of ACM, April 2004, Vol. 47, No. 4.

[6] Gopal, A, Mukhopadhay, T, Krishnan, M, “The Role of Software Processes and Communication in Offshore Software development”, Comm. Of ACM, April 2002, Vol. 45, No. 4

[7] Carmel, E., ”Global Software teams”, Prentice hall, New Jersey, 1999
[8] SEGAL Global SW Engineering lab in the Computer Science Department of University of Victoria, BC http://segal.cs.uvic.ca/csc576b/
[9] Aspray W., Mayadas F., Vardi M.Y., Editors: “Globalization and Offshoring of Software, A Report of the ACM Job Migration Task Force”, ACM 2006, http://www.acm.org/globalizationreport/

[10] Olly Gotel, Christelle Scharff, Sopheap Seng, “Preparing Computer Science Students for Global Software Development”; 36th ASEE/IEEE Frontiers in Education Conference, Oct. 2006, San Diego
[11] Wing H. Huen,” An Enterprise Perspective of Software Offshoring” 36th ASEE/IEEE Frontiers in Education Conference, Oct. 2006, San Diego

[12] Dragutin Petkovic, Rainer Todtenhoefer, Gary Thompson: “Teaching Practical Software Engineering and Global Software Engineering: Case Study and Recommendations”, 36th ASEE/IEEE Frontiers in Education Conference, October, 2006, San Diego, CA.

[13] D. Petkovic, G. Thompson, R. Todtenhoefer: "Teaching Practical Software Engineering and Global Software Engineering: Evaluation and Comparison", The Eleventh Annual Conference on

Innovation and Technology in Computer Science Education, University of Bologna, Italy 26-28 June 2006

[14] Martin Grimheden, H. F. Machiel Van der Loos, Helen L. Chen, David M Cannon and Larry Leifer, “Culture Coaching: A Model for Facilitating Globally Distributed Collaboration”,. 36th ASEE/IEEE Frontiers in Education Conference, Oct. 2006, San Diego

[15] Jerry Boetje, Foundational Actions: “Teaching Software Engineering When Time Is Tight”, The Eleventh Annual Conference on Innovation and Technology in Computer Science Education, University of Bologna, Italy 26-28 June 2006

[16] N. Levitt: “The Changing World of Outsourcing”, IEEE Computer, December 2007

[17] D. Petkovic, G. Thompson, R. Todtenhoefer: "Assessment and Comparison of Local and Global SW Engineering Practices in a Classroom Setting **",** The Thirteenth Annual Conference on Innovation and Technology in Computer Science Education, Madrid, Spain, June 2008

[18] D. Petkovic, G. Thompson, R. Todtenhoefer, S. Heang : “Analysis of Global SW Engineering Practices in a Classroom Setting”, Computer Supported Cooperative Work, November 2008