2009

E-Learning For Software Engineering: A Case Study On Teaching Information Systems Online Group Project With Extreme Programming

Burcu Bulgurcu
The University of British Columbia, bulgurcu@bc.edu

E-LEARNING FOR SOFTWARE ENGINEERING: A CASE STUDY ON TEACHING INFORMATION SYSTEMS ONLINE GROUP PROJECT WITH EXTREME PROGRAMMING

Burcu Bulgurcu
Department of Management Information Systems
The University of British Columbia
burcu.bulgurcu@sauder.ubc.ca

Abstract:

This paper reports the experience gained in software engineering group work within the framework of a fourteen week master’s level graduate course on information systems development. Teams of three to five members developed web-based application systems using the Distributed eXtreme Programming (XP) methodology. A case study is proposed to understand the issues encountered by students during the software development process and to determine the impact of XP methodology as well as team members’ geographical distribution on students’ overall performance. We suggest that teaching an information systems group project online with agile methodology (i.e. distributed XP) brings several issues to be considered before and during the development process. This study points out these issues, particularly those regarding student teams’ communication, coordination, and collaboration practices. Improvement of these issues in the future would help educators develop more effective education settings and help students enhance their performance.

Keywords: distributed software development, virtual teams, e-learning, software engineering, extreme programming

I. INTRODUCTION

Over the last decade, parallel to the consistent development of information technologies and globalization of the world, the phenomena of distributed software development and virtual teams have been intensified. As a result, increasingly dispersed development practices across geographical distances were introduced. Even though distributed software development is a highly challenging, complex and risky process, today’s competitive business environment requires multi-site, and even multinational and multicultural development settings for many software development projects [Herbsleb and Moitra, 2001]. However, distributed software development is a fairly new phenomenon, which is still evolving and needs further research and work to understand and cope with these challenges and complexities. Likewise, virtual teams represent a new form of organization that offers flexibility and responsiveness and has the potential to revolutionize the workplace [Powell et al., 2004]. While traditional software development offers valuable theoretical background and a starting point, virtual teams with their unique material, technical and social challenges call for additional, specialized research to enhance their effectiveness [Powell et al., 2004]. Despite the availability of the necessary information communication technology infrastructures, effective utilization of these technologies for distributed development has not yet been clearly discussed.

This paper presents an empirical study which attempts to address and discuss some of the significant questions on virtual software development in an educational context. Accordingly, the development processes and the projects of five different virtual teams of an online software development class were investigated. The goal of this study was to observe the ad-hoc development processes of virtual software development teams, especially their communication, coordination, collaboration, and group decision making practices and to report the major issues of the educational setting as well as the issues of the software development based on the student teams’ experiences. Further goal of the study was to improve the development processes of the teams by utilizing an online collaboration support tool, called XPWeb, which is designed to support the distributed extreme programming projects. The software development methodology adopted for the course was extreme programming (XP). The fundamental principles of XP, including test-driven and iterative development practices, continual communication with the customer, and collective ownership of developed software and refactoring were fully adopted in this course.

The case study focuses on the following research question: What are the major issues that relate to *communication, coordination,* and *collaboration* practices of software development teams when team members are geographically separated from each other?

II. TERMINOLOGY

Distributed Software Development

A team is defined as a collection of individuals who are interdependent in their tasks but share a common goal and responsibilities for targeted outcomes [Cohen and Baily, 1997]. The concept of virtual teams is defined as; *"groups of geographically, organizationally and/or time dispersed workers brought together by information and telecommunication technologies to accomplish one or more organizational tasks"* [Alavi and Yoo, 1997]. Distinctive features of virtual teams from the traditional ones are basically their reliance on information technologies to communicate with each other, their flexible composition, and their ability to traverse traditional organizational boundaries and time constraints [Powell et al., 2004]. Distributed software development is defined as the software development activities disturbed across multiple sites [Mockus and Herbsleb, 2001].

Distributed software development and virtual teams are known to create both unique advantages and challenges [Herbsleb and Grinter, 1999; Mockus and Weiss, 2001]. Overcoming the distances and extending the possibilities of partnerships are usually key motivations for distributed software development. Additionally, economic forces have turned national markets into global markets and introduced new forms of competition as well as cooperation that reach across national boundaries [Herbsleb and Moitra 2001]. Distributed development, however, also introduces a number of significant challenges for knowledge management, communication, coordination, and process management. In particular, geographically distributed development teams face enormous communication and coordination problems. A case study demonstrates that the difficulty of initiating contact with other partners or team members and several potential unanticipated events can stretch project communication to the breaking point, and project schedules can become compromised as a result of problems in project integration and coordination [Herbsleb and Grinter, 1999]. Lack of trust between the sites, problems in relationship building, and the lack of cohesion are other unique challenges of the distributed development [Powell et al., 2004; Herbsleb and Grinter, 1999; Mockus and Weiss, 2001].

Extreme Programming (XP)

XP is one of the most popular agile software development methods that has been proven to be successful at many organizations functioning in different business contexts [Extreme Programming, 2009]. XP aims at a customer oriented software development methodology which stresses customer satisfaction. The requirements are prioritized according to customers' needs. Instead of attempting to develop all possible requirements in one shot in the requirements analysis phase, as conducted in traditional software development methodologies, XP aims to deliver the software as needed by the customer. XP also empowers software developers to confidently respond to changing customer requirements even at the later stages of development life cycle. To ensure customers' satisfaction and understand their needs, developers have constant communication with their customers as well as fellow programmers. Developers receive ongoing feedback not only by communicating with their customers but also by testing the developed software starting from the beginning of development. Delivering completed part of the system to customers as early as possible, having their feedback, and implementing suggested changes are essentials of the development cycle. Development is iterative and success depends on the unique contributions of team members as well as customers. Communication is one of the essential elements of XP. Simplicity, feedback, respect, and courage are the other essentials that are expected to improve the performance and the quality of the software project. XP aims to implement a simple, yet effective environment to enable team productivity and efficiency. XP enables developers to respond to changing requirements and technology with confidence [Beck and Andres 2004; Beck and Fowler, 2000; Jeffries et al. 2000].

The planning phase is one of the most important phases of XP development. This phase includes development of *user stories* by the customers. User stories represent usage needs of the system users and are used to create time estimates by system developers. Instead of having large requirements documents, user stories include about three-to-four sentences to explain user needs. Next, *release plans* are created to estimate each user story in terms of ideal programming weeks by the development team. It is important to include all the stakeholders' views in the release plan so that technical people and business people can have a negotiated schedule that everyone can commit to. In the next phase, the development team releases iterative versions of the system to the customers. Iterative development adds agility to the development process. Iteration planning often includes identifying tasks, tracking down the process, and re-estimations. Iterations are expected to end with a testable software product. Having customer available is one of the most important requirements of XP, as they are expected to help developers with time estimates, priority assignments, and unit testing. Unit tests are created before coding so that developer is usually clearer about the expectations of the coding. Moreover, because each code piece is followed by a unit test, developer has immediate feedback while working. Lastly, the overall system is tested according to acceptance tests which are created from user stories. Each acceptance test represents expected results from the system. Customers are responsible for verifying and reviewing test scores to decide which failed tests are of highest priority [Extreme Programming, 2009].

III. TEACHING CASE

The Course and the Participants

Participants of this study were the students of Informatics Online (ION) 505 course in the Informatics Institute of Middle East Technical University (METU). Informatics-Online (ION) [ION, 2009] is the fully Internet based master's degree program of the Information Systems department of the Informatics Institute. All courses of ION program are carried out asynchronously over an e-learning system; called Netclass. The objective and the program structure of ION program are specified as providing expertise on the fundamental and current concepts of information technology and systems to working professionals who need continuing education without the need to come to the campus for lectures.

The ION 505 course, named Information Systems On-Line Group Project, aims to integrate fundamental awareness and expertise obtained in earlier information systems and software engineering courses into a meaningful whole through an applied group project. Course material, in parallel with project stages, presents an overview of fundamental IS project planning and management techniques and approaches. Classroom and Internet-based versions of the course have been started in the Informatics Institute of the same university in 1997. The waterfall approach has been utilized in these courses until 2003. XP methodology has first been introduced in the classroom course (SM 504) in the 2003 fall term, and then in the Internet-based course (ION 505) in the 2004 spring term.

The author of this paper was the research assistant of the ION 505 course during the term in which the study was conducted. She has actively participated in and observed all class and team discussions through the forums of the online class, and joined online team meetings and e-mail lists of project groups. The instructor of the course has been teaching the master's level software engineering group project courses in the Electrical and Electronics Engineering Department (EE 647) and Informatics Institute (SM 504, ION 505) of the METU since the late 1980s.

The participants of the case study were twenty-five students of ION 505. Students were all Turkish, from different geographical locations (twelve of the students were located in Ankara, seven were in Istanbul, one was in Hakkari, one was in Antalya, three were in Europe, and one was in Russia). The online project teams consisted of three or four students. At least one of the group members was geographically separated from others. The teams reported that even the members who were located in the same city could not hold face-to-face meetings but had to work separately and communicate over information communication technologies. Moreover, the instructor of the course who acted as the customer of the development projects was geographically separated from all student teams. Students were expected to select projects that were sufficiently extensive in scope to experience the challenges of the requirements development process and understand the unique challenges of inner and outer group communication of distributed development. Nevertheless, they were only expected to develop prototypes of projects along with a little coding. The following projects were developed by the development teams:

Team 1: Library Automation System

Team 2: Scholarship Management System

Team 3: Online Education Framework

Team 4: Online PC Information Database

Team 5: Debt Automation System

Team 6: Business to Business (B2B) e-Marketplace Framework

The Support Tool

To enhance team members' planning, communication and coordination practices and to overcome the obstacles of their geographically separation, a support tool called XPWeb [XPWeb, 2006] was provided to all teams. XPWeb was a simple web based tool which was designed to manage the projects developed with distributed XP methodology. The tool provided four major functionalities:

1) *Planning section* is used to prepare the project schedule and release plan, create user stories and tasks, discuss the stories with the customer, select user stories and tasks, assign team members to tasks, and define roles and responsibilities of group members.
2) *Calendar section* is used visualize the tasks of weeks. Task information includes task definitions, assigned team members, and due dates.
3) *Reports section* is used to report and assess the performance of the developers and the development processes of development iterations and to provide feedback to team members and the customer about the progress.
4) *Versioning section* is used to store and download the latest version of the developed system. End products are made available to group members and the customer.

Software Development Method

The method utilized for software development was distributed XP. Besides implementing the iterative cycles of XP, teams were also required to submit formal documents with IEEE standards including Software Requirements Specification (SRS - IEEE 830-1998), Software Design Description (SDD - IEEE 1016-1998) and Software Test Documentation (IEEE 829-1998) in each development cycle. Teams updated their initial documents according to the proposed iteration plans of each cycle and created the documents iteratively. At the end of the fourth cycle, all teams were expected to submit one SRS, one SDD, and one acceptance test report in addition to the running software. The class started with two weeks of introduction. These weeks involved the following procedures:

1. The instructor introduced himself and communicated the objectives and requirements of the course. He informed students about the schedule and the development methodology. He also asked students to introduce themselves and explain their educational and professional backgrounds and interests. He then asked them to propose their preferred team roles for the development project and the tasks that they choose to handle.
2. The instructor required the students to form groups of 3 to 4 people. In the meantime, he asked instructed them to learn the basics of XP and its base practices from his suggested references.
3. Students formed their project groups and submitted a project proposal.
4. The forums for the communication of all groups were created in Netclass system. Students were required to employ the forums for inner group communications and for their decision making activities.
5. The instructor revised the proposals and asked for the release plans.
6. The instructor introduced the support tool, namely; XPWeb. He asked students to read the lecture notes on the utilization and effective usage of the tool. Students were required to employ the support tool during the development. The support tool not only helped students with their development activities, such as scheduling, planning, reporting, and versioning, but also helped the instructor (and the researcher) to follow up the end-products each week. Furthermore, the tool helped the instructor (and the researcher) examine the roles and responsibilities of team members.
7. Students presented the project plans which were developed according to the guidelines of XP. Students developed user stories, user validation criteria, requirement priorities, their weights and implementation risks in that phase. In the later phases, the instructor acted as the customer of the development project.

After the introduction weeks, students were involved in project implementation cycles. There were four iterative cycles, each lasting for two weeks. Iterative cycles involved four fundamental phases:

System Requirement Analysis Phase: Teams developed and submitted their iteration plans. Based on their release plans, they selected critical stories to be implemented in the following iteration cycle. They updated the weight, priority, and risk indexes of their stories from their previous plans. They created the tasks for the selected stories and assigned the tasks to team members. The deliverables of this phase were the iteration plan and the updated SRS report.

System Design Phase: Teams developed their design activities according to the user stories of their iteration plans. The deliverable of the phase was the updated SDD report.

Systems Implementation Phase: Teams developed the running software system according to the selected stories. They updated their initial plans in the support tool, and reported the team performance as well as the individual performances. The deliverable of the phase was the running software system and the code of the system.

Testing Phase: Teams developed the unit tests for testing. They validated the running software system with the predefined validation criteria. The deliverables of the phase were the debugged software system and the updated acceptance test report.

IV. METHOD AND DATA COLLECTION

Multiple data collection methods are typically employed in case studies. Ideally, evidence form two or more sources converge to ensure reliability of the research findings [Benbasat et al., 1987]. Utilization of multiple sources of data also supports triangulation (cross-validation of information and conclusions through the use of multiple procedures of sources). Thus, we utilized multiple data collection methods as the data collection strategy. Three main data collection methods—interviews, direct observations, and artifact and document analysis—were employed. The summary of the data collection procedures is as follows:

- The researcher directly observed the development processes of the teams during the education term as the teaching assistant of the course. Direct observations included observations and analysis the end products of each iteration cycle as well as inner group communications (by joining the e-mail groups, following the class discussions on the e-learning webpage, analyzing the logs of XPWeb etc.).
- An online survey was conducted at the end of the semester to understand students' general perceptions on the development setting.
- Group interviews were conducted with the student teams.

V. FINDINGS OF THE STUDY

The major issues of the educational setting and teams' software development processes were grouped under four categories as follows:

1. The issues regarding the educational setting: Some of the major issues were directly related to the characteristics of the educational setting, as the participants of the study were students of an online course.

Time restrictions: Team members reported that despite the importance of analysis phase, allocated time was very limited, which resulted in negative impacts on the development of group dynamics, performance of group decision making, and definition of clear roles and responsibilities. Team members suggested that because of their geographical separation, formation of a group culture and dynamics were essential for better performance. Teams argued that initialization phase should be strongly encouraged by the instructor and supported by several team formation activities.

Teams suggested that, in particular, they suffered from the following problems due to time restrictions:

- *Issues regarding team work:* Teams could not establish a team culture, which resulted in diverse work focuses of team members, lack of motivation and discipline, and unbalanced work load.
- *Issues regarding overall planning activities:* Teams could not propose accurate size and feasibility estimations for user stories in the analysis phase. As a result, they frequently had to deviate from the initial plan during the development.
- *Issues regarding definitions for roles and responsibilities***:** Teams could not propose clear definitions for team members' roles and responsibilities. The lack of definitions resulted in ineffective inner group communication and coordination practices since clear role and responsibility definitions were essential to establish the communication protocols and to outline the activity based management plan.

Difficulty in accessing the customer: Since the projects were developed with the XP methodology, teams frequently needed to access to the customer. However, they argued that they had difficulties when they tried to communicate with the customer.

Difficulty in improving the development processes: Teams suggested that they could not improve their development processes among the iteration cycles due to time restrictions, even though they thought improvement was necessary.

Need for a comprehensive tool support: Teams argued that even though they had benefited from the support tool; XPWeb, they needed a more comprehensive communication infrastructure. They explained that even though XPWeb was adequate for planning purposes, it lacked necessary communication and coordination interfaces and decision making features. Since the tool did not satisfy the demands of the teams, they had to utilize other communication tools, such as instant messaging or e-mail. In general, teams used the forum of the class website for asynchronous communication and XPWeb for planning purposes, and used their private e-mails and instant messaging tools for synchronous communication, particularly for inner group communication and decision making purposes. In particular, they used the private e-mails and instant messaging tools for inner group communication and decision making purposes. However, using different tools created problems, such as, communication inefficiencies, information and end-product loses, and difficulties in keeping track of the conversations.

2. The issues regarding teams' communication practices: Teams explained that their ineffective communication practices introduced the most difficult problems to overcome. The communication needs were explained as follows:

Need for a communication plan: Teams explained that they needed a comprehensive communication plan that would specify all the needs of team communication, such as selection of communication media according to the purposes of communication, specification of meeting schedules and available working hours of team members. Teams added that protocols of communication should be defined in the early phases of development. The protocols should clearly inform the team members about the means of communication, their expected utilization, allowed response times for any communication requests, and strategies for managing the knowledge accumulated by team communications. Teams also argued that communication with the customer should be improved for better requirement elicitation. Thus, the communication plan should also incorporate the communication schedules with the customer as much as possible.

Need for a communication leader: Teams argued that in any distributed development setting, there should be a communication leader to coordinate the communication traffic. The communication leader should not only have the responsibility of teams' information transfer, but also work as a team leader and take initiative as a decision maker when necessary.

Need to support synchronous communication: Teams explained that they preferred synchronous communication over asynchronous communication because of its effectiveness and efficiency advantages. However, because teams did not know of other team members' work schedules, they had to communicate asynchronously (i.e. e-mails) most of the time, which resulted in inefficiencies in inner group communications. Teams provided two main explanations for the inefficiencies:

- *Misunderstandings in asynchronous communication:* One of the team members explained their problem as follows: *"The written communications often caused misunderstandings which resulted in more serious problems. For example, two of our team members were assigned implementation and testing tasks. However, due to frequent communication problems, we had to assign both tasks to one member and this strengthened the unbalanced workload among our team members."*
- *Delays in completion of work tasks:* Because it was difficult to predict when the message will be read and responded to, asynchronous communication usually resulted in delays in the completion of work processes. Team members explained that when they attempted to communicate over an asynchronous communication medium, they usually utilized another synchronous communication tool, such as telephone, to confirm whether the message was received by the other team member.

3. The issues regarding teams' coordination practices: Teams explained that they had several coordination problems as a result of their ineffectiveness in their planning, teamwork, and communication practices. The coordination needs were explained as follows:

Need to enhance parallel work and reduce unbalanced work load: Teams explained that most of the tasks were highly dependent on each other or had to be completed in sequence. So, teams could not manage to work on the same task as a group or to process several tasks in parallel. Ineffectiveness of communication practices also had negative impacts on coordination of sequential tasks. The problem particularly resulted in unbalanced workloads for some of the team members. For example, one of teams explained that they had two experienced coders in the team, so they assigned the coding tasks to both. However, due to their communication problems, one of the coders had to take the full responsibility of the entire coding task. Teams also mentioned that each team member had to strictly commit to the deadlines of the tasks for which he or she was responsible, because the iteration cycles were very short and the minor delays of each of the team members usually resulted in major delays at the end of the development cycles.

4. The issues regarding teams' collaboration practices: Teams explained that they had several collaboration problems because they could not develop a team culture. The teams' collaboration needs were explained as follows:

Need to agree on a common goal: Team members' lacking of a common goal and their having distinct work focuses and motivations were proposed as the major causes of collaboration problems.

Need to enhance group decision making: The team's ability to perform group decision making was proposed as the most significant factor to enhance group collaboration. It helped team members clarify their roles and responsibilities and enhanced their involvement and in turn their motivation to work as a group.

VI. CONCLUDING REMARKS

The use of virtual teams in software development is a promising but challenging phenomenon. This study aimed to reveal major issues of distributed software development and the effectiveness of virtual teams in an educational setting. The main conclusions drawn from the study are as follows:

- Having *strong communication infrastructures* is one of the major success factors of virtual development. The need for a *comprehensive communication plan and a communication leader* is strongly emphasized. According the findings of the case, the communication plan should specify: *the communication schedule, the media to contact other team members, required response time to reply communication requests, points of contact for team members.*
- Having *clear definitions for team members' roles and responsibilities* is one of the major success factors of virtual development. Lack of clear definitions introduced several communication, coordination, and collaboration problems such as: difficulties in developing the communication plan, unbalanced information distribution among team members, unbalanced work load, dependencies on specific team members, delays in sequential tasks and work processes, distrust of other team members.
- Having a *comprehensive web-based tool support* is one of the major success factors of virtual development. The tool should be comprehensive and suitable for the team structure, development methodology, and development processes. Furthermore, teams should be trained about the features of the tool and usefulness of the system. Lastly, teams should be given the necessary adaptation time to get accustomed to the tool.
- Having a *strong team culture* that would provide a focus of work is one of the major success factors of virtual development. Results of the case study suggest that sufficient amount of time should be devoted to team formation processes in the project initialization

phase. The instructor should encourage student teams get know each other to build up a team culture. Team members should have good knowledge about other team members' work styles, available project working hours etc.

- Having *strong coordination practices* is one of the major success factors of virtual development. Based on this case study, the following issues were understood to be vital for the success of team members' effective coordination:
  - Work tasks should be clearly defined. The tasks should be well-managed and distributed to team members to support parallel work as much as possible.
  - The work flows should be clearly defined and synchronized by considering the sequence of events, task and actor dependencies, work schedules, and available resources. The parallel work of team members would reduce idle times of team members and help synchronize the end work products.
  - Strong communication practices would also help prevent problems of ineffective coordination.
  - Processes involving geographically-separated team members should be planned to require as minimum contact as possible.

This study discussed the issues of students in an information systems online group project. However, the question of how we can mitigate these issues still needs further discussion. Further research needs to propose solutions to resolve the highlighted issues. Another area of research may be the implementation of a comprehensive support tool that reflects the identified needs of the development teams.

The findings of this study were based on an educational setting and concern the issues encountered by students in a short period of time. However, distributed software development is known to be a much more complex phenomenon, particularly in the form of virtual organizations and global development, because of the differences in developers' cultural backgrounds and/or spoken languages. Hence, this study simply aims to provide a broad understanding about the issues encountered by the student development teams. Generalization of findings to other contexts should be done with careful attention. A fruitful research direction could be to compare the findings of this study with those of the studies investigating global organizations functioning in different contextual settings.

VII. REFERENCES

Alavi, M. and Yoo, Y. (1997). “Is Learning in Virtual Teams Real?”, *Working Paper Harvard Business School*, MA.

Benbasat, I., Goldstein, D.K. and Mead, M. (1987). “The Case Research Strategy in Studies of Information Systems”, *MIS Quarterly*, (11)3, pp. 396-368.

Beck, K., Andres, C. (2004). *Extreme Programming Explained: Embrace Change (2nd Edition)*, Addison-Wesley Professional.

Beck , K., Fowler, M. (2000). *Planning Extreme Programming*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA.

Cohen, S. G. and Bailey, D.E. (1997). “What Makes Teams Work: Group Effectiveness Research from the Shop Floor to the Executive Suite”, *Journal of Management, 23(*3), pp. 239-290.

Extreme Programming (2009, September 28). Retrieved October 26, 2009, from Extreme Programming: A gentle introduction website: http://www.extremeprogramming.org/

Herbsleb, J. and Grinter, R. (1999). “Architectures, Coordination, and Distance: Conway’s Law and Beyond”, *IEEE Software, 16(*5), pp. 63–70.

Herbsleb, J. and Moitra, D. (2001). “Global Software Development”, *IEEE Software, 18(2),* pp. 16-20.

ION, Informatics Online (2009). Retrieved October 26, 2009, from Informatics Online website: http://ion.ii.metu.edu.tr

Jeffries, R. E., Anderson, A., Hendrickson, C. (2000). *Extreme Programming Installed*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA.

Powell, A., Piccoli, G. and Ives, B. (2004). “Virtual Teams: A Review of Current Literature and Directions for Further Research”, *The DATA BASE for Advances in Information Systems, 35(1), pp. 7*.

Mockus, A. and Herbsleb, J. (2001). “Challenges of Global Software Development”, *Seventh International Software Metrics Symposium (METRICS’01)*, pp. 182-184.

Mockus, A. and Weiss, D. (2001). “Globalization by Chunking: A quantitative approach”, *IEEE Software*, 18(2), pp. 30.

XPWeb (2006). Retrieved October 26, 2009, from XPWeb Source Forge website: http://xpweb.sourceforge.net/

VIII. ABOUT THE AUTHOR

Burcu Bulgurcu is a PhD student at Sauder School of Business, the University of British Columbia. She holds a bachelor’s degree in Computer Education and Instructional Technologies from Middle East Technical University, a master’s degree in Information Systems from Middle East Technical University, and a second master’s degree in Management Information Systems from Sauder School Business, the University of British Columbia. Her research interests include instructional technologies, global software development, and human and organizational aspects of information security and privacy.