Students as Partners and Students as Mentors: An Educational Model for Quality Assurance in Global Software Development

Olly Gotel[1], Vidya Kulkarni[2], Christelle Scharff[1], Longchrea Neak[3]

[1] Pace University, Seidenberg School of Computer Science and Information Systems, New York, NY, USA
{ogotel, cscharff}@pace.edu
[2] University of Delhi, Computer Science Department, New Delhi, India
vkulkarni@cs.du.ac.in
[3] Institute of Technology of Cambodia, Computer Science Department, Phnom Penh, Cambodia
longchrea.neak@itc.edu.kh

Abstract. Since 2005, Pace University in New York City has been collaborating with the Institute of Technology of Cambodia and the University of Delhi in India to bring students together to work on globally distributed software development projects. Over this period, we have been exploring models through which graduates and undergraduates from the three countries can work together as partners and mentors, with pedagogical value to all sides. In 2007, we converged on using the critical activity of Software Quality Assurance as a focal point around which to establish a partnering and mentoring relationship. We included seven students from the Masters Program in Software Design and Engineering at Pace University to help assure the quality of what was to be a single distributed project involving twenty-seven students from across three global locations. A team of these graduates acted as internal quality assurance mentors for the US undergraduates and another team acted as external quality assurance auditors for the overall project. To further focus on quality, requirements and testing activities were emphasized throughout the duration of the project. The motivation, logistics and experiences from this project are reported in this paper, and lessons of wider applicability to educational and industrial practice are provided.

Keywords: *Auditing, Global Software Development, Mentoring, Requirements, Software Engineering Education, Software Quality Assurance, Testing.*

1 Introduction

It has become increasingly common to set up a Global Software Development (GSD) experience for students as part of their Software Engineering training in an undergraduate Computer Science degree [5, 12, 16]. There are also many outstanding research challenges with global settings for Software Engineering that educational experiences can contribute towards understanding [13]. The initial time a student project of this nature is undertaken by collaborating institutions, there is a steep learning curve for the instructors involved, and a number of constraints and logistical steps can easily be overlooked as assumptions are made about distant locales and remote educational practices [10]. The first year is much about discovering a model for working that fits all the parties, both instructors and students, spanning time zones, place and cultures. Such learning experiences can help provide for a smoother second year, in turn allowing for the exploration of particular Software Engineering concerns, practices or working arrangements amongst the students. For example, outsourcing the development of a well-defined software component and then integrating it into an evolving software system [8]. Necessarily, the overall model for collaboration still develops as new information is uncovered, but an agreed and repeatable approach for running global projects can gradually emerge.

Having undertaken GSD projects for two years, involving instructors and students from across three global institutions, we made the observation that it becomes all too common to focus time and effort on the logistics of the project, meaning that the timely completion of some form of software product by the end of the project becomes more central than the quality of what is produced. It can be expedient for busy instructors, as well as rewarding for overwhelmed students, to see something demonstrable, whatever the quality. Engineered for a snapshot in time to achieve a grade, little consideration gets given to actual deployment and longevity.

Quality has been defined in numerous ways in the literature. Two of the more prevailing definitions are those of Crosby and Juran: *"conformance to requirements"* [4] and *"fitness for use"* [15], as reflected in the ISO 9000 family of standards for quality management systems [14]. In the context of software development, quality is generally measured in terms of the specified requirements that are satisfied by the software system that is produced. According to the proponents of Software Process Improvement (SPI), it has long been argued that one of the most effective ways to achieve software quality is to adopt and follow a suitably mature software process [1]. Software Quality Assurance (SQA) is, at a fundamental level, all those process-related activities that are undertaken in the pursuit of achieving software quality. SQA involves requirements, design and code reviews, requirements and defect tracking, testing (both validation and verification), and much more. Such activities are undertaken to monitor and manage the software development process and its products to help ensure that any problems are addressed early on.

Given that undergraduate Software Engineering is typically a student's first exposure to software development processes, practices and principles, concepts are frequently taught in the classroom just as they need to be applied to the practical project the students are working on. This affords little opportunity for students to step back and examine their work from the broader and more objective perspective that is required for SQA. It also prevents them from having the requisite oversight at the onset of the project to plan for SQA appropriately. As a consequence, SQA is an activity that we have found difficult to prepare our undergraduate students for, particularly since it can be perceived to get in the way of the 'real' task of developing something demonstrable by the course deadline. It has thus been a challenge for us when running our GSD projects for students, projects that are already far more complex than single site co-located projects, to find an active and student-led way to focus on the very critical and professional topic of SQA.

For our third year of institutional collaboration, we therefore deemed it essential to focus on injecting a concern for quality into the GSD project. To account for some of the issues noted above, we decided to engage Software Design and Engineering graduate students in both internal and external SQA capacities. The internal role was to help undergraduates focus on the important quality practices related to requirements and testing, and the external role was to monitor and provide feedback on the end-to-end process and its deliverables. Moreover, by getting the students to work together on a single project that we anticipated would be deployed in Cambodia, we expected the need for quality to be clearly recognized.

This paper describes the third year project experience and its results. In Section 2, we provide the background and objectives for this work, also summarizing the institutional collaboration to date that we are building upon. We give details of the project arrangements for this third year in Section 3. In Sections 4 through 6, we describe the emphasis that was placed on requirements and testing, mentoring and auditing respectively. Our findings are presented in Section 7 and wider lessons are drawn with respect to the role of partners and mentors in software development endeavors in Section 8, especially when involving new students and recent hires. We finish the paper with conclusions and a synopsis of our ongoing work.

2 Background and Objectives

Undergraduate Computer Science students at Pace University in the US have been collaborating with undergraduate Computer Science students from the Institute of Technology of Cambodia (ITC) for three years. This collaboration takes the form of an annual GSD project for their capstone Software Engineering courses. Within these projects, the Cambodian students typically act as clients and testers for the project, and the US students typically act as developers. This permits the students to experience a reversal of traditional offshore outsourcing software development roles and thus gain first hand experience of the needs of the stakeholder roles that they are likely to collaborate with when they work in an industrial setting. Over the past two years of the collaboration, this arrangement has also involved graduate Computer Science students from the University of Delhi in India playing a more recognized role for Indian practitioners of offshore third-party service providers.

Throughout the three years of the collaboration, we have been examining the nature of the roles that students will play in the global market place, including the entrepreneurial opportunities for technologists, in order to prepare students with the requisite skills and sensitivities. Also, we have been exploring the processes and communication models that are useful to employ in this context, along with the tools that can be used to support them. These findings have been reported in previous papers [8, 9, 10].

This current paper describes the spring 2007 GSD project where students from the three countries worked together on a single project with the intention of developing a software system to be deployed into operation within Cambodia. The focus was on scaling up and so on the true need for both local and global integration, along with an emphasis on requirements and testing for quality. This project incorporated a competitive bidding process for an outsourced component of the work to future maximize quality efforts, and made use of US graduate students who were specializing in Software Engineering to act in either an internal or external capacity to help assure the quality of the distributed global project. The website of the 2007 project is available at *http://atlantis.seidenberg.pace.edu/wiki/gsd2007*.

2.1 Research and Teaching Objectives

This study was designed to build upon the objectives and findings from the two previous years of running GSD projects for students. The two specific objectives for this study were as follows:

- To provide a ‘real’ project that would require students to take software through to production quality. This was to enable students to learn about ‘whole-life’ software development and so the ‘total cost of ownership’ of software systems. In addition, we wanted to start to reinforce longevity in our institutional relationships and so provide the potential for future student teams to play a role maintaining a legacy system. The primary focus was therefore for students to learn about what it takes to engineer a sustainable quality software system.
- To investigate how to incorporate students with specialist skills into the educational model so as to: (a) relieve the instructors of some of the day-to-day work spent on attempting to manage and check the quality of the students’ work; and (b) address questions and provide timely advice to the undergraduate students, as and when needed outside of the classroom setting, in a ‘safe’ environment.

3 Project Context

We addressed these objectives by focusing on a single project that was important to the Cambodian school involved in the study and intended to be deployed into operations in Cambodia. Students were to work in global sub-teams on separate components of a larger project that would demand integration. In addition, the project was to incorporate a competitive bidding process for a well-defined component of the work in an attempt to enhance quality through design diversity. Software Engineering and Design graduate students were recruited as ‘specialists’ and given loose guidelines to mentor and audit the US undergraduates. This section provides information about the participating institutions, the students, the project and the team set-up. It also clarifies the logistics and technologies associated with running a project of this nature.

3.1 Collaborators and Courses

The Institute of Technology of Cambodia (ITC) (*http://www.itc.edu.kh*) is a leading semi-public higher education school in Phnom Penh. The study described in this paper targeted the Software Engineering course for fourth year ITC Computer Science undergraduate students.

Pace University (*http://www.pace.edu*) is a private university located in New York. The study described in this paper targeted the capstone undergraduate Software Engineering course taken by junior (third year) and senior (fourth year) Computer Science students in New York City. It also involved students from the Masters Program in Software Design and Engineering who were concurrently taking a Software Reliability and Quality Assurance course. Approximately half of the graduate students had a professional career in the New York City area software industry.

The University of Delhi (*http://www.du.ac.in*) is one of the prestigious public institutions in India. The study described in this paper involved the second year Master of Computer Applications students studying a Database Applications course.

3.2 Student Roles and Responsibilities

Cambodian Students are Clients and Testers. Their responsibilities were to work directly with the US undergraduate students to describe the software they wanted to be built and the context in which it was to operate. They therefore had to review and give feedback on the requirements, design and testing documents, test the produced software system and submit bug reports. At the end of the semester, the Cambodian students had to assess the software system developed by the US students (with Indian sub-contracting) and to compare this with the software system developed solely by the Indian students (see Indian roles later in this section). In addition, the Cambodian students were to prototype their own version of the software system from the requirements that had been jointly developed, to satisfy an additional course requirement imposed by the Cambodian instructor.

US Undergraduate Students are Developers and Lead Contractors. Their responsibilities were to elicit the requirements from the clients and produce an agreed requirements specification, propose design options that subcontract part of the system design and development to Indian students, prepare a Request For Proposal (RFP) to send to Indian students, receive and evaluate responses to the RFP, implement the software system and test it, while concurrently handling requirements changes, integration of the outsourced component, accounting for feedback and managing the end-to-end contract. At the end of the semester, the US students had to deliver the software system to their clients.

US Graduate Students are Internal SQA Mentors and External SQA Auditors. The mentoring role was designed to help ensure that the US undergraduate students were doing what they needed to do to assure a quality result, and so graduate students were charged to help them perfect the techniques and practices introduced in the classroom, especially with respect to writing requirements, tracing them through to design and code, document versioning, change management techniques, integration, and test planning and execution. Mentors were to act as internal eyes for the project and issue periodic reports to the instructors (as partners). The auditing responsibility was orchestrated to review the artifacts delivered by the undergraduates and the processes used to deliver them. This involved checking for conformance to documented processes, verifying whether the specified requirements were satisfied in design and code, and determining the quality of the testing activities. Auditors were to act as the quality gatekeepers in the project and deliver periodic audit reports to the instructors and to the SQA mentors (as partners).

Indian Students are Third-party Suppliers. Their responsibilities were to submit separate bids for the outsourced component and then collaborate on the selected bid. In response to a requirements specification, they were to provide the US undergraduate students with a database design and the corresponding SQL code, with sample data to be integrated in the overall system design. In addition, and outside the initial intent of the project, the Indian students decided to develop their own variant of the software from the full requirements specification in competition with the US undergraduate student work. Note that this effort was undertaken with visibility of the evolving requirements provided on a shared wiki, but without direct communication with the Cambodian clients.

3.3 Project MultiLIB Description

ITC is composed of six departments, including the Department of Computer Science. Besides a central library, most departments have their own internal library. The internal library of the Department of Computer Science provides many resources to its students, such as books in English and French, student internship reports, CD-ROMs, DVDs and e-books. To date, the library management tasks and other related transactions have not been computerized. The departmental secretary, acting as the librarian and using an Excel spreadsheet, currently manages the records of all the resources available in the library. In order to make the students aware of the available resources in the library there is a paper list pasted in front of the office of the department. Not only is this unwieldy as the resources increase in number, it is never clear what is currently available for loan and what is not. The objective of this project, called MultiLIB by the Cambodian instructor, was to develop a multi-purpose web-based library management system to be used by students, professors and the secretary / librarian.

The development effort was to be partitioned in the following way (also a designated requirement of the Cambodian instructor):

- *Librarian / Administrator Side* – To focus on that part of MultiLIB that will be used by the librarian and administrator for managing the library policies, all the resources, the accounts in the system and all the issue / return transactions.
- *Guest / Student / Professor Side* – To focus on that part of MultiLIB that will be used by guests, students and professors to view the information on the available and new resources in the library, to reserve, rate and recommend the resources, and to consult the status of their personal accounts in the library (for students and professors only).
- *Innovation Side* – To focus on that part of MultiLIB that will be used by students to view the electronic resources, such as e-books, audio and video. Moreover, this side was intended to be forward looking and so to consider future innovative features likely to be required of MultiLIB as it evolved.

The innovation side of MultiLIB was not intended for the initial release of the software. This meant that the other two sides needed to be engineered to account for future anticipated extensions and integration needs.

3.4 Teams

In spring 2007, the undergraduate class at Pace University comprised eight students, the class at ITC comprised thirteen students and six students were drawn from the thirty-student class at the University of Delhi. Seven Pace University graduate students also participated in the project.

The US undergraduates were split into two sub-teams of four students each, one team to work on the librarian / administrator side of the project (henceforth referred to as the librarian side of the project in this paper) and the other team to work on the guest / student / professor side (henceforth referred to as the student side).

The Cambodian students were split into three sub-teams, five to work on the librarian side, four to work on the student side, and four to work on the innovation side. The Cambodian teams were to consider the requirements for that aspect of MultiLIB and to work closely with the US sub-teams that would be writing the requirements and subsequently developing them, thus working as globally extended sub-teams. Further, since the librarian and student sides had to integrate to give a complete system, both global sub-teams had to liaise with each other and with the innovation side (based only in Cambodia) to account for future requirements when designing the architecture. Each of the local sub-teams had a leader, along with documentation, communication and quality assurance managers.

The Indian students were organized initially as three sub-teams of two students each. Each sub-team had to bid for the database work. The winning design was then to be developed by the full set of six students. This set of students also went on to develop their parallel version of the full software system out of their own volition, as highlighted earlier.

One US graduate student was assigned to each US undergraduate sub-team to act as its mentor. A third graduate student was selected as a floating mentor for both sub-teams, mostly to assist with the integration aspects of the project. The three mentors were chosen to be that subset of the graduate students who were also currently working in the software industry full-time whilst taking their masters degree. The remaining four graduate students acted as auditors. They were organized into pairs and assigned to the two US sub-teams to monitor and report on their progress.

In summary, the global teams were composed of US undergraduates, US graduates, Cambodian and Indian students, as illustrated in Figure 1 and detailed in Table 1.

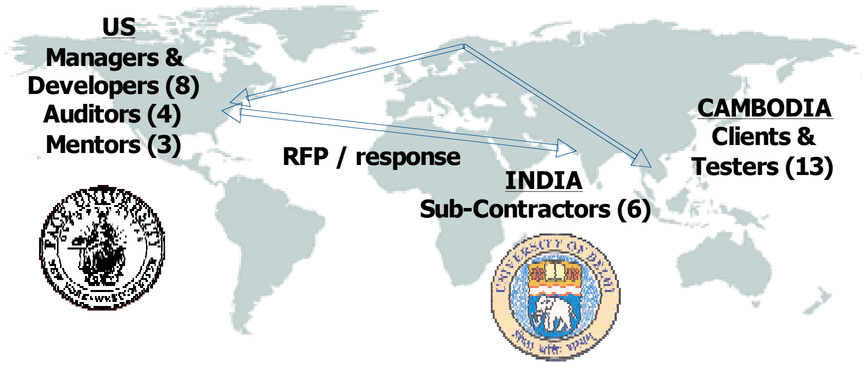

Fig 1. Global and Local Working Arrangements on Project MultiLIB.

Table 1. Team Set-up on Project MultiLIB.

Sub-team	**Roles and Numbers**	**Location**	**Additional Team Members**
Librarian Side	5 Clients	ITC, Cambodia	1 Integration Mentor - Pace Graduate and Industry Professional, NYC 6 Sub-contractors (Developers) – University of Delhi Graduates, India
	4 Developers	Pace Undergraduates, NYC	
	1 Mentor	Pace Graduate and Industry Professional, NYC	
	2 Auditors	Pace Graduates, NYC	
Student Side	4 Clients	ITC, Cambodia	
	4 Developers	Pace Undergraduates, NYC	
	1 Mentor	Pace Graduate and Industry Professional, NYC	
	2 Auditors	Pace Graduates, New York	
Innovation Side	4 Clients	ITC, Cambodia	

3.5 Logistics: Process, Technology and Communication Tools

The project milestones were organized around one week for initialization of the project and team bonding, five weeks for requirements, three weeks for design, three weeks for coding / construction and two weeks for testing. The Software Engineering process model that was used was a loose waterfall model with iteration and feedback cycles, for instructor control and visibility. This was a first course in Software Engineering for all the undergraduates involved, so the class sessions and practical project work needed to be aligned. In the requirements phase, requirements were captured using questionnaires and chat discussions. Templates were designed for the requirements, design and testing documents to help the students standardize their work. Open source development and communication tools were used to account for radical differences in the assumptions that could be made about software and Internet availability in the three countries. The technologies used in this project are discussed in a companion paper [9].

4 Requirements and Testing for SQA

Central to any definition of 'quality' is the term 'requirements'. Our Software Engineering courses therefore pay much attention to techniques that can be used for gathering stakeholder requirements, particularly across distances, and for communicating this understanding of the requirements across languages and cultures in a manner that promotes feedback. Requirements are also the basis for testing, without which assessments as to the attained quality cannot be determined. A second emphasis in our courses is therefore on bringing testing activities forward in the software development lifecycle to associate them with requirements and so more consciously work toward their satisfaction. This section describes the efforts undertaken in these two areas to stress a discipline for SQA in the MultiLIB project.

4.1 Requirements Process

Typically for our students, the requirements process goes through the phases of feasibility study, requirements elicitation, documentation and validation. At the beginning of the project, the US undergraduate students received a two-page description of MultiLIB and a PowerPoint presentation describing its context and features, as prepared by the Cambodian instructor. They also received samples of the current Excel spreadsheets used by the librarian to deal with library loans and the management of available resources. Students used these documents as a starting point to determine the requirements of MultiLIB. They then refined and added requirements gathered through questionnaires sent to the Cambodian students, followed-up with chat sessions, and gathered during a face-to-face session with the Cambodian instructor when he visited the US. Diversity in the channels for gathering requirements were important due to the difficulties students had to understand each other, mainly due to language and terminology issues (English is the third language of the Cambodian students, after Khmer and French). The requirements were specified using a simple template for consistency that articulated the users, the source of the requirements, priority and how to test the requirement. Validation took place in chat sessions and using a validation checklist that requested the Cambodian students to accept, accept with modification or reject each requirement. The validation activity permitted the students to go to the next phase of the software development lifecycle with more confidence in the requirements.

4.2 Testing Process

Some integration testing of the software system took place during the development process, but most of the testing that took place was system-level and user acceptance testing, as these activities were emphasized during the two weeks before the delivery of the software system. System testing and user acceptance testing were to be undertaken by the Cambodian members of the team since they were the owners of the requirements. The US developers prepared a testing document for the Cambodian testing team. This document contained all the data necessary to access the software system (e.g., URLs and login data), along with details of sample books stored within the database, the list of implemented requirements with the name of the developer responsible for each requirement, and the testing scenarios used for manual testing of each of the functional requirements. The Cambodian students used this testing document in conjunction with the requirements document to test the software.

The Indian students prepared a sixteen-page user manual for their variant of the software system. This included descriptions of all functionality included in the software system, along with the unique features provided, and also user-ids and passwords for testing the website. Sample books and reports were already inserted into tables for testing purposes at the end of the user manual. The same set of Cambodian students performed system testing and user acceptance testing on the Indian version.

4.3 Bug Reporting and Issue Tracking

The Issue Tracker facility of *java.net* was used by the Cambodian clients and US developers to report, fix and manage bugs throughout the project. Issue Tracker offers facilities to enter, view and modify an issue, query and track issues, and to generate status reports about issues. Use of the facility enables students to understand the lifecycle of an issue and the different states issues can pass through. It also allows the generation of testing reports, detailing open and closed issues, and high importance issues at a given moment in time, helping to visualize the evolution of and ensure the quality of the software product. During the testing phase, each issue submitted by the Cambodian testing team was assigned to the US team member in charge of the development of the feature associated with the raised issue, and broadcast to the whole development team. Java.net was used by the client mostly to report bugs in the software rather than by the developers to eliminate them. The latter was due to a lack of time and the fact that testing took place only during the two last weeks of the project when developers were still developing and testing the software in an interleaved manner.

The Indian students did not use a specific tool for their testing. Each and every page in their developed web-based system was separately tested with test data during the development and then the complete system was tested with the test data. When the Cambodian students tested the whole system, they sent emails to one of the Indian students who manually undertook issue tracking and resolution.

At the end of the semester, the Cambodian students provided one testing summary of the software developed by the US team and by the Indian team, comprising the number of issues identified in each, along with a description of the main issues discovered. The Cambodian students used this as a basis to compare the two software systems and to provide a final assessment as to the software system they would be accepting and why.

5 Mentoring for SQA

Mentoring is: *"The offering of advice, information, or guidance by a person with useful experience, skills, or expertise for another individual's personal and professional development"* [11]. A 'mentor' is: *"A wise and trusted counselor or teacher"* [6]. In this project, US graduate students who were specializing in Software Engineering and working in the software industry full-time were tasked with mentoring local undergraduate students.

The role of internal SQA mentor was therefore established to provide a point of contact and support structure for the two undergraduate sub-teams based in New York. The two components of MultiLIB (the student side and the librarian side) were to be developed independently by the sub-teams, but needed to fit together. Accordingly, one graduate student was assigned to each sub-team to act as an advisor and mentor, while a third graduate student was assigned to an overarching role to mentor both sub-teams, keeping an eye on the integration aspects of the project, whilst also standing in for the other mentors if needed.

The intention of the mentoring role was not for graduates to do the work for the undergraduate students, but to help ensure that the students were doing what they needed to do to assure a quality result. Also, to help the students perfect some of the techniques and practices introduced in class, especially with respect to writing requirements, tracing them through to design and code, document versioning and change management techniques, and test planning and execution. Specifically, they were to work directly with the project leaders of the sub-teams to help them with their management tasks and in resolving some of the typical student difficulties faced with team working. For the instructors, the mentors were to act as internal eyes for the project.

The mentors were tasked with preparing an SQA plan describing what they were proposing to do as a team of mentors to help the students assure the quality of their work. This included details of the activities, processes, methods, standards, etc. they considered relevant, along with a timeline and communication strategy to engage with the undergraduates. Each mentor was to maintain an individual log of what they did with the students and why, along with the outcome of any advice provided or sessions they ran. They were specifically tasked to keep track of what they considered effective or not on the project in terms of their interventions. The mentoring team produced a final report summarizing their experience, lessons and recommendations for future projects.

6 Auditing for SQA

Auditing is an important and often regulatory activity undertaken in many domains (e.g., financial audit and environmental audit). It is an activity undertaken both internally by organizations and externally by engaging independent third parties. It is not solely about compliance, checking that an organization is doing what it is supposed to be doing, and so offering reassurance to its customers, but also about alerting to any required intervention before it is too late to be effective. Auditing activities commonly rely upon interviews with personnel and examination of documentary evidence.

The role of external SQA auditor on this project was to take a more objective look at the quality of the US undergraduate students' contributions to the global project. Since the undergraduate students were to be preparing documents (in draft and final form) at certain key dates throughout the project, the auditors' responsibilities were to review everything that the students were doing (both the products they were delivering and the processes they were using to deliver them) and to provide a periodic audit report. This was to involve checking for conformance to documented processes, verifying whether the specified requirements have been satisfied in design and code, and determining the quality of the testing (amongst other activities). They were encouraged to do most of their work face-to-face with the undergraduate students. The audit report was to be provided to the instructors (as their clients and partners) and to the SQA mentors (as their partners). The latter was to enable their peers to deliver feedback to the undergraduate students and to determine what activities they may need to focus on going forwards. All feedback was to be constructive, suggesting actions to be taken if weaknesses or deficiencies were found. The auditors were to work together as the quality gatekeepers for the global project but, unlike the mentors, they were not to be part of the global project team as such. They were required to maintain their distance and objectivity to look at what the students were doing without an invested stake.

The auditors were tasked with preparing an SQA audit plan describing what they were proposing to do as a team of external auditors to assess and help assure the quality of the student work. This was to include details of all the activities they were to undertake, how they proposed to undertake them, along with a timeline and forms / templates they considered useful. Each auditor was to keep an individual log of what they did with the undergraduate students and why. The team was to provide a final report on the quality of the completed work (process and product) and to justify their conclusions.

7 Findings

In this section, we summarize some of the key observations and findings with respect to this additional emphasis on quality-related activities in our MultiLIB GSD project experience. These findings relate to the quality that was obtained on the project, the impact of a focus on requirements and testing, and the value of mentoring and auditing.

7.1 Overall Quality Level

The final version of the requirements specification included thirty-four functional requirements and eleven non-functional requirements. The US students implemented eighteen functional and three non-functional requirements in the delivered software system. The non-functional requirements that were implemented related to technical constraints (e.g., implementation language, database and browser types), while more complex requirements associated with performance and security were neither implemented nor tested. In contrast, the Indian students implemented twenty-eight functional requirements and four non-functional requirements. They concluded that some requirements were either not feasible or implemented as a by-product of technology choices, so not in need of special action, and there were a small number of requirements they attempted but did not fully implement. Based on java.net Issue Tracker, thirty-nine issues were submitted by the Cambodian students for the US software system and forty-seven issues were submitted for the Indian software system, half of these issues being defects and half of them being requests for enhancement. The Cambodian students rejected both of the software systems that were developed for them.

While the Indian version of the software implemented more features (i.e., satisfied more of the requirements), it was also delivered with more outstanding issues. However, it was considered to have a "*more attractive user interface*" and to be more secure, which led to perceptions of higher quality, and hence the Cambodian students preferred the Indian version. For example, the Indian software generated alerts for invalid inputs whereas the US software generated none. Note that the Indian students had no contact with the clients and were working directly from the requirements specification that was developed and continuously evolved by the US students. This is an interesting finding as it provides some evidence that a well-written requirements document can communicate across many dimensions of distance. The Indian students reported that the main problem they faced, other than a lack of time to complete the work, was that many times they did not get replies to their queries from the US students. They understood that this was because the US students could not get their queries resolved by the Cambodian students in a timely fashion also. Time delays got perpetuated along the communications chain.

The main issues with the US students' software were related to the fact that, due to time constraints, they only implemented a very simplified version of the software to speed up implementation and deliver something of value. They eased their development task by making assumptions, such as a book can have only one author and cannot contain subtitles. Moreover, features that were specified as high priority for the users by the clients did not gain sufficient attention and produced incorrect results when implemented, which frustrated the clients. The US students spent lots of their time improving the quality of the requirements and design documents, and keeping them up to date on the wiki, diligence and effort that the Indian students benefited from. Focusing on requirements and their versioning meant that development and testing time for their own implementation was squeezed.

It should also be noted that the Indian students regarded the development of this software as a professional challenge -- they were competing with the US students and dedicated all their time to the development of this software and continued to do so for up to two months after their classes had ended, whereas the US students focused more on the Software Engineering practices they were learning and their application, and ceased activities when the semester ended.

7.2 Requirements and Testing Focus

All students were new to the type of tool used in this project for bug reporting and issue tracking. The Cambodian students found java.net a difficult and heavy tool to use, but very powerful in terms of describing issues in detail. The US and Indian students both recognized the importance of the tool, especially for facilitating communication between the development and testing teams, though the Indian students preferred a non-tool-oriented approach. However, a recurrent problem was that the bugs reported by the clients were not always perceived as bugs by the developers. For instance, the developers did not realize that some of the software issues that they considered unimportant, especially those raised during the non-functional testing, were crucial to the clients and for the deployment of the software system. In some cases, the identified bugs in the Indian software were rectified, but in most cases the Indian students replied that the issue raised (or feature mentioned) was not a part of requirements document. They often felt that the clients were asking for new features after the testing activity and raising these requests as issues. The perception of the Indian students was that the clients only realized they should have asked for additional features when they started testing the software.

The Cambodian students were not satisfied with the usability of either the US or the Indian software. For instance, they found that the web interface was not easy to navigate and not uniform from one page to the other. Additionally, the web components were not positioned the right way on the web page. Trying to defend the software they developed, both the US and Indian students felt that the Cambodian students gave "*too much attention on little things instead of the main functionalities*". By 'little things' they meant usability issues, one of the main issues contributing to software failure [7, 17]. They also thought that: "*things like invalid input tests were useful but should not have been something that was heavily concentrated on (most bug reports were about invalid outputs)*". Further, Indian students felt that some of the issues mentioned by Cambodian students after testing, relating to the screen design, layouts and the application not performing as expected, were not properly specified in the requirements document and, as such, could not be expected. These common problems, cited in [3], were directly experienced by students and are generally attributed to communication challenges. Working with a client made students eventually realize the importance of addressing and satisfying non-functional requirements like usability.

7.3 Mentoring Activities

The mentors ran bi-weekly and then weekly meetings with the undergraduates as they decided they needed to be more pro-active in their efforts. These meetings were mostly conducted with the sub-team project leaders, the maximum group attendance only ever being nine out of a possible eleven. These meetings focused on the following activities:

- *Requirements Engineering* - Validation of the template used to gather requirements and suggestions as to additional information to capture to facilitate traceability; checking the techniques used for gathering and validating requirements; ensuring that traceability is established between requirements and software components; examining the version control method used; and-suggesting methods to help ease the identification of changes made to requirements.
- *Project Scheduling* - Assistance in developing a project plan for the entire project; ensuring the milestones are aggressive enough to meet deadlines; and internally auditing the execution of the project plan to verify if milestones are achieved.
- *Technical Approach* - Suggesting design standards and guidelines to follow; initiating brainstorming sessions to develop the designs; and looking into approaches to prepare for integration.
- *Construction* - Suggesting test-driven development techniques and tools; verifying that the software component version control strategy; and encouraging the team to include a maintenance and deployment strategy for their product early on.

In the words of the mentors: *“At these meetings we went over their progress. At one meeting we went over the requirements documents for each team, page by page, and offered suggestions.”*

The mentors reported that the technical ability of the undergraduates was always lagging behind where it needed to be to complete the project. By adding more interaction into the equation for the undergraduates, via mentors and auditors, this had the perception of slowing them down even further. As one graduate reported: *“By the time we evaluated the problem and made suggestions, too much time had gone by and they had had to figure it out without our help.”* This is a direct consequence of the frequent need to teach Software Engineering theory concurrently with its practice for the first time. Also, it implies that the mentoring needed to have been initiated earlier -- it did not start for a few weeks due to class schedules and prior commitments, so it was always running in catch-up mode.

While they made themselves available to the undergraduates, the mentors felt that the undergraduates didn’t really always know they needed help or were too busy to seek it. The graduates therefore found themselves having to balance being pro-active in their role with not intervening and taking over. They reported, in retrospect, that the most valuable thing they offered may have been simply: “*giving the undergraduates a pat on the back when needed, a gentle push towards the goal, and a vision of where to go and what to do to achieve it*”. Also, sharing experiences from their various workplaces about what can go wrong if they don’t do some of the things they have been asked to do on the project was considered valuable. From the undergraduate perspective they reported that: *“it was nice to have these mentors there just in case everything decided to fall apart.”*

7.4 Auditing Activities

The two audit teams created and shared a template to use to run face-to-face interview sessions with the undergraduates. The audit template implemented a ‘traffic light’ system for SQA where a list of criteria to examine were provided and the result of the audit would be categorized as either red (non-compliant or not doing), green (compliant or doing) or amber (some issues). Each team reviewed core milestone documents of the two sub-teams and undertook four audits, which they reported on. Typical checklist items in the audit included:

- *Team and Communications* - Are they organized? Do they communicate well with the Cambodian team? Is there a defined purpose and scope for the project? Does the team know its roles? Any foreseeable problems? Solutions?
- *Requirements* **-** Is there a template or standard being used? Is it being properly implemented? Is there a unique identifier for each requirement? Are the requirements feasible? Are the requirements prioritized? How are the requirements prioritized? Do they list constraints and assumptions? Have they been verified? Have they walked through their requirements document? What validation techniques were used? Are the requirements acceptable to the client? Is there any versioning control in place?
- *Design, Code and Test* - Does the design correlate with the requirements? Is the code following a standard? Is the code well commented? Is there a bug report / tracking procedure implemented? Do the tests make sense? Do they adequately test the requirements? Are they being carried out?

Given the fact that it took time for the auditors to receive and then review the project artifacts, schedule interviews with undergraduates and subsequently write their audit reports, it would be a number of weeks before the feedback got back to the instructors, mentors and students. This was not as effective as it should have been. Auditing, without the ability to receive and respond to the audit feedback in a timely fashion, is of little constructive value. This is another consequence of the reduced cycle times of student projects and concurrent learning activities. However, some issues were identified, allowing mitigating actions to be recommended, and open issues could be tracked and re-visited at consecutive audits. In this way, the auditors provided a valuable set of external eyes for the project.

From a technical perspective, the auditors uncovered unrealistic prioritizations, assumptions and missing test cases in the requirements when reviewing the documents. From a social perspective, they noticed when the teams were fragmenting and losing spirit. Sample technical issues from their audit reports include: (i) *"There was a mistake in Requirements R8 and R10 but they were cosmetic mislabeling that will hopefully be corrected quickly. NFR5 doesn't have a test case, we have recommended that they look at it and try to do something with it."* (ii) *"The code isn't really following a template but it seems to be relatively well organized. We did try to emphasize the need for better commenting of the code because there is difficulty in understanding what it does without it."* (iii) *"The Testing Stage is pending and whether it will truly be experienced by the team due to the deadline is questionable. The Test Specification is unavailable and pending completion at this time."* Sample communication issues listed in their audit reports include: (i) *"There is a problem stemming from an idea that the Student's software won't be implemented, either through miscommunication or a misunderstanding this has lead to a huge blow to the motivation/morale of the team as a whole. The project has become a chore to them because its relevance has been destroyed." (ii) "The team seems to have improved, the communication between members and between international teams has become more consistent, although there is a bit of lag in terms of some paper updates."*

Despite some of the issues with timeliness in the implementation of this model, show-stopping crises were avoided and the other students participating in the project were keen to have their work audited also. Notably, and in the spirit of learning and improvement, the Indian students requested an audit. The auditors themselves gained a valuable learning experience: *"The GSD project gave me a chance to approach a process that was rarely shined upon in my undergraduate studies. It gave me a chance to develop a procedure that I believe is very successful in garnering information from my auditees."*

8 Lessons and Recommendations

The original research and teaching objectives were to find a way to emphasize the need for quality in a student GSD project and to seek a way to share responsibilities for accomplishing this on a project of this nature via the students themselves. We attempted to achieve the former by instigating a real project to be integrated, deployed and maintained and to achieve the latter by setting up a network of mentors and auditors.

We believe we developed an innovative working relationship between instructors, graduates and undergraduates in this project, and satisfied the second objective through a model that we are now refining in our latest GSD student project work. Although the software system was not completed and not accepted, we do not believe this negates the benefits that all parties obtained from mentoring and auditing. The lack of client acceptance of MultiLIB was not due to a failure in the concept per se, but more an issue associated with its implementation for the first occasion. Every time a project setting is changed there is an entirely new learning process to go through and fine-tuning the arrangement can take time. This outcome does mean that we did not fully succeed in realizing the first objective though. While quality was improved with respect to requirements and designs, in the opinion of all the instructors when comparing with previous efforts, this was not the case for the final software system. The project schedule was perhaps too heavily biased towards the upstream software development activities. Our lessons and recommendations for others are provided below.

8.1 Focus on the Partnerships in GSD

A 'partner' is defined as: *"One that is united or associated with another or others in an activity or a sphere of common interest"* [6]. In this study, students from across the globe and across degree levels partnered with each other, and graduate students also partnered with instructors. These are rare partnerships to foster in student-based project settings, particularly to entrust senior students with critical roles, but are models of working that we recommend others explore.

8.1.1 Establish Student Partnerships

Undergraduate and graduate Pace University, ITC and University of Delhi students partnered to work on one project. The advantages of this model of education include:

- *Reciprocal Learning* – Each party in the global project had distinct backgrounds, skills and perspectives on the project, allowing them to learn from each other. For example, the US undergraduates made friends with the Cambodians and learned about their country, culture and technology situation. Conversely, the US students exchanged information on the role and use of wiki technology in American society and in professional business settings, a technology that was new to the Cambodians. In addition, the US undergraduates and graduates got to know each other, something that does not ordinarily happen give day and night time scheduling in New York.

The former gained a support network and benefited from practical experience-based explanations that contextualized Software Engineering theory, and the latter felt empowered from the ability to practice Software Engineering activities at a different level (they also identified some candidate employees).

- *Accountability* – Both the US and Indian students felt accountable to the Cambodians for delivering a working software system. This motivated their work efforts. When rumors spread that the software would not be used (see Section 7.4), this had devastating effects on morale until remedied. Creating a self-styled competitive situation was what motivated the Indian students since they did not have the direct obligation to the Cambodian students. Real Projects for Real Clients Courses (RPRCC) is a growing movement in academic settings and we suggest that GSD projects are the ultimate way for students to experience these [2].

Some outstanding issues were identified and need to be addressed by others seeking to adopt such an educational model:

- *Contribution Disparity* – As with many student projects, working in a team setting can present more difficulties that working with new processes and technologies. It can often result in some students carrying the burden of the work, individuals believing that they know best and overriding team decisions and, in due course, loosing friends. Given the work demanded on a global project of this nature, all these issues are more prominent and were all experienced in this project. While equally experienced in industrial settings, student project leaders have little option to make firing and hiring decisions. Perhaps a closer simulation of industrial settings needs to be explored as a way to remedy this, else individual assessment structures need more attention.
- *Global Team Unity* – Due to the project set-up, the Indian students never felt entirely part of the global team; they were service providers. This led to a competitive streak and the request to be audited. Where the requirements custodians are in competition with those developing the requirements (the US students who were writing the requirements were also developing them in competition with the Indian students), this makes for an imbalance in the incentive for cooperation. While no negative repercussions were experienced during the MultiLIB project, this is an area to pay attention to. Competing student teams need to be on a level-playing field, just as one would expect fairness in industrial competitive arrangements.
- *Coordination* – Shared awareness and the exchange of project artifacts were facilitated through the use of wikis on this project. However, when wikis were not updated in a timely manner, or when students became too focused on development to look at the wiki, coordination problems resulted. For instance, it was not until the Indian students were about to release their software to the Cambodians for testing that they realized that the requirements document they were working to was not the one that was posted as current. Expecting beleaguered students to poll for important change information is unrealistic. As in an industrial setting, such information needs to be pushed to relevant parties and change control processes need to be established to promote a less accidental way of monitoring change.

8.1.2 Establish Partnerships Between Instructors and Students

Pace University graduate students partnered with instructors. Getting senior students and instructors to work together as peers on a project is recommended for the following reasons:

- *Delegation and Oversight* – Broader visibility of the project, facilitated by regular reports on progress from both the mentors and auditors, meant that a set of graduates were able to provide a completely different perspective on the work. This enabled the instructors to delegate some of the overall management task. Managing a global project is often the 'hidden' cost in GSD arrangements and something that can easily detract instructors from venturing down this educational path. We suggest that a carefully constructed partnership model, engaging students who are ready to put some of their skills into practice, is one was to alleviate this burden.
- *Timely Intervention* – The mentors helped to uncover technical training needs and team management skills that were required but not supplied in the curriculum, and they were able to directly address many of these day-to-day issues. The auditors provided much broader alerts to systemic project issues, giving the opportunity to intervene where necessary, including issues that influence team spirit and jeopardize the project. The value of a multi-layered educational model is exactly this breadth and depth of perspective that is gained.
- *Improved Quality of the Requirements and Design* – With the intense focus on the writing and reviewing of the requirements, and of the participation of the mentors in helping the undergraduates to architect the system and the auditors in providing feedback, the quality of both the requirements and design improved. The graduates helped to convey the importance of quality to the undergraduates outside of the class setting, as well as assist students in their practices, thereby reinforcing the instructors' class instruction.

However, there are some issues that deserve attention when instituting such an arrangement:

- *Insufficient Audit Planning and Expectations* – The audit planning started too late in the project to allow for adequate review cycles, as time was needed to prepare suitable audit checklists. This meant that audits were often undertaken too late. Audits need to be planned early, sufficient time needs to be factored in for reporting and responding to the audit, and contingency is necessary to accommodate delays. Even though graduate students were undertaking this role, most had never played such a role on a project and needed more guidance than expected to extrapolate from their own project experiences to create guidelines and checklists to examine the work of others. It would help to share such checklists with other instructors.
- *Inadequate Cycle Time for Feedback* – The feedback to instructors, and so to the undergraduate students, was not always timely. Working by day, taking classes by night and meeting in class once a week does not put the project at the top of a graduate's priority list. When a semester is only a small number of weeks, every lost week is a huge percentage of the overall timeframe. Such issues could be addressed by building the students' contributions much more significantly into the grading scheme for their class because time cannot be extended.
- *Sustaining a Quality Focus* – The quality of the US software system was not as high as had been expected following on from their requirements and design work. While the Indian students were able to leverage some of the improvement in quality of these other artifacts, the US students ran out of time and energy, and the auditing also became squeezed downstream. A far greater proportion of the time needs to be scheduled to accomplish closure in SQA efforts – quality is not guaranteed from improved requirements alone and this attention must be sustained.

8.2 Institute Mentoring Networks in GSD

In the words of the final report from the mentors on this project: *"Mentoring is not intended to develop a narrow set of skills, but instead to develop the whole person toward advancement in his/her career. Mentoring supports individual development through both career and psychosocial functions."* In many organizations, mentoring is undertaken to develop assets, help retention and transfer knowledge [11]. We suggest that the complexity and sensitivity of GSD projects should leverage this approach to institutional learning and development to help get new students through particular tasks.

It is worth mentioning that the mentors on the MultiLIB project described what they believed they did as shorter-term 'coaching'. We suggest that the advantages of using this approach in GSD educational initiatives, whatever the label, are as follows:

- *Goal Setting* – Breaking down and planning tasks is a fundamental project activity, yet it is difficult for instructors to sit down and create detailed plans to direct all students. This is an ideal role for graduate students who should have been through the process many times and equally an opportunity for them to develop their professional skills.
- *Provide Technical Training* – It is not possible for instructors to cover every topic that the students may need to use on a software development project. A network of 'experts' can augment the learning experience and customize specialist training to individual needs. Although co-located in MultiLIB, we are exploring distributed mentoring and coaching in our ongoing work, a model that industrial organizations could equally benefit from.
- *Confidante for Team Leaders* – Learning the realities of working as a team, whilst learning about software development and striving for grades, can be stressful. Students acting as team leaders are often placed in unfamiliar and uncomfortable situations and they need to know there is someone other than the instructor they can turn to to explain problems and gain advice. Where do project leaders of global projects turn when the stress builds? One would expect them to be more seasoned, but there is an obvious need to examine the support available for trainees.
- *Provide Rationale and Explain Consequences* – By sharing corporate experiences, graduates were able to motivate and reinforce information provided in class, contextualizing many things for the students. In all projects endeavors, unless activities are perceived to add value there can be resistance to their implementation. Taking time to explain, especially by non-judgmental external parties, pays off.
- *Professional Development* – The students undertaking the mentoring gain the opportunity to play reciprocal and new roles. *"And it was interesting for me to be involved with a project at a high level rather than doing the coding myself. I think if I had this opportunity again, I could do a better job as mentor to a group of developers."* All parties engaged in software development have personal training needs and these can potentially be fulfilled through project support roles of this type.

There are a few issues that need to be considered when creating a network of support for an educational project, some of which are essential for the perception of 'fairness':

- *Even Participation* – Mentoring needs to be for everyone in the project, else students feel alienated. On MultiLIB, there was the observation that the mentoring mainly benefited the team leaders, and this impacted team cohesion. Mentoring needs to be set up in a more balanced manner so that all students gain direct value.

- *Undergraduates were Uncomfortable being Proactive* – The mentors in this project initiated all communications: *"If the students did not, in the end, need us, then we cannot fault them for not asking for help. If they did need us, but did not know they did, then we did not do a good enough job at setting up our relationship vis-à-vis the requirements for the course."* The lesson here is the need to explain more carefully the role of mentors in terms of the help they can provide and the way to go about seeking this help. Those able to benefit from support must feel empowered to use a resource that has been established for their benefit without fear or judgment.

8.3 Summary

While the result of having introduced mentoring and auditing was not as profound as we had anticipated on this project, we certainly saw evidence as to the benefits of orchestrating multiple levels of partnership amongst students and instructors on student software development projects. The structure provided via mentors is a model that extends the reach of the classroom sessions and serves to augment the skills and knowledge base of those directly engaged in the project work. The activities undertaken in the form of auditing can improve quality if executed in a timely fashion and if sufficient time is scheduled to address audit findings, but these activities need to be sustained throughout development. Mentoring and auditing is one way to actively spotlight the requirements and testing activities that are so central to software quality, and are highly recommended to other educators initiating GSD student projects, as well as a model to promote for wider industry practice.

9 Conclusions and Ongoing Work

This paper reported on a continuing GSD educational initiative between the US, Cambodia and India. In the words of one of the students in the post project survey: *"From the interaction with Pace University students, I learnt that the basic thought process of students from anywhere is same. This project was a true example of globalization."* Indian students, more directly touched by GSD in their daily lives than the Cambodian students, were particularly thankful for the experience and asked for certificates of their participation in the project to give to their future employers. Further, every single student of the second year Master of Computer Applications program in India was offered a job one year before graduation. In the post project survey, one of these students qualified the experience as a "*golden opportunity*" and a US student noted it as a "*great learning experience that consumed [her] life this semester*".

During this third year of the collaboration, we created a model through which international undergraduate and graduate students could work together as partners and mentors on a GSD project. We also revealed how this model that promotes internal and external SQA, along with a competitive bidding exercise, led to improved quality in the written requirements and the design, the latter somewhat due to the ability to leverage the variability in the three design bids. Our work in 2008 is focusing on some of the outstanding issues with this model, notably ensuring that sufficient time is spent on realizing the requirements in a working system, delivering quality in software and not just in documentation, whilst involving all the students from across the globe in the mentoring and auditing arrangement.

Acknowledgments. This work is supported by a National Collegiate Inventors and Innovators Alliance (NCIIA) Course and Program Development grant (#3465-06), "Incubating the Next Generation of Global Software Development Entrepreneurs" (2006-2008). We are thankful to all the students of Pace University, the Institute of Technology of Cambodia and the University of Delhi who were involved in this project.

References

1. Ahern, D. M., Clouse, A. and Turner, R. *"CMMI Distilled: A Practical Introduction to Integrated Process Improvement"*. 2nd Edition. Addison-Wesley, 2003.
2. Almstrum, V., Condly, S., Johnson, A., Klappholz, D., Modesitt, K. and Owen, C. "A Framework for Success in Real Projects for Real Clients Courses." In Ellis, H., Demurjian, S. and Naveda, F. (Eds.) *"Software Engineering: Effective Teaching and Learning Approaches and Practices"*. Hershey, PA: IGI Global, 2008.
3. Aloi, M. and Fortin, W. "Utilizing IBM Rational Tools to Successfully Outsource in a Globally Distributed Development Environment". In *IBM Rational Software Development Conference*, 2007.
4. Crosby, P. B. *"Quality is Free"*. Signet, 1980.
5. Damian, D., Hadwin, A. and Al-Ani, B. "Instructional Design and Assessment Strategies for Teaching Global Software Development: A Framework". In *Proceedings of the 28th International Conference on Software Engineering (ICSE 2006)*, Shanghai, China, 20-28 May, 2006, pp.685–690.
6. Farlex, Inc. *"The Free Dictionary"*. http://www.thefreedictionary.com/, 2008.
7. Foraker Design. *"Usability First: Your Online Guide to Usability Resources"*. http://www.usabilityfirst.com/intro/index.txl, 2002-2006.
8. Gotel, O., Kulkarni, V., Neak, L., Scharff, C. and Seng, S. "Introducing Global Supply Chains into Software Engineering Education". In *Proceedings of the 1st International Conference on Software Engineering Approaches For Offshore and Outsourced Development (SEAFOOD 2007)*, Zurich, Switzerland, 5-6 February, 2007.
9. Gotel, O., Kulkarni, V., Neak, L. and Scharff, C. "Working Across Borders: Overcoming Culturally-Based Technology Challenges in Student Global Software Development". In *Proceedings of the 21st Conference on Software Engineering Education and Training (CSEET 2008)*, Charleston, South Carolina, USA, 14-17 April, 2008 (to appear).
10. Gotel, O., Scharff, C. and Seng, S. "Preparing Computer Science Students for Global Software Development". In *Proceedings of the 36th ASEE/IEEE Frontiers in Education Conference (FIE 2006)*, San Diego, USA, 2006.
11. Harvard Business School Press. *"Coaching and Mentoring: How to Develop Top Talent and Achieve Stronger Performance"*. September, 2004, Page 76.
12. Hawthorne, M. J. and Perry, D. E. "Software Engineering Education in the Era of Outsourcing, Distributed Development Distributed Development, and Open Source Software: Challenges and Opportunities". In *Proceedings of the 27th International Conference on Software Engineering (ICSE 2005)*, St. Louis, Missouri, USA, 15-21 May, 2005, pp.643–644.
13. Herbsleb, J. D. "Global Software Engineering: The Future of Socio-technical Coordination". In *Proceedings of the 29th International Conference on Software Engineering – The Future of Software Engineering (ICSE-FASE 2007)*, Minneapolis, Minnesota, USA, 20-26 May, 2007.
14. The International Organization for Standardization. *"ISO 9000 - Quality Management"*. The ISO Standards Collection, 2007 (ISBN 978-92-67-10455-3).
15. Juran, J. M. *"Juran's Quality Control Handbook",* Gryna, F. M. (Ed.). 4th Edition. McGraw-Hill, 1988.
16. Richardson, I., Milewski, E., Keil, P. and Mullick, N. "Distributed Development – an Education Perspective on the Global Studio Project". In *Proceedings of the 28th International Conference on Software Engineering*, Shanghai, China, 20-28 May, 2006, Pages 679 – 684.
17. Rideout, T. B., Uyeda, K. M. and Williams, E. L. **"**Evolving the software usability engineering process at Hewlett-Packard"**.** In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, 14-17 November, 1989, Pages 229 – 234 (Volume 1).