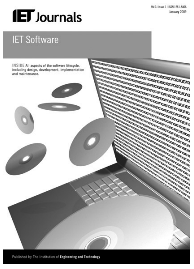
Published in IET Software
Received on 1st May 2011
Revised on 1st October 2011
doi: 10.1049/iet-sen.2011.0078

Special Section: Management of Global Software Development: Opportunities, Challenges and Lessons Learned



ISSN 1751-8806

Mixing continents, competences and roles: five years of lessons for software engineering education

O. Gotel[1] *C. Scharff*[2] *V. Kulkarni*[3]
[1]*Independent Researcher, New York, NY, USA*
[2]*Seidenberg School of Computer Science and Information Systems, Pace University, New York, NY, USA*
[3]*Department of Computer Science, University of Delhi, Delhi, India*
E-mail: olly@gotel.net; cscharff@pace.edu

Abstract: This paper describes five years of a software engineering education initiative led by Pace University in New York City. The initiative brought together faculty and students from across the globe to work on distributed software development projects, encompassing the Institute of Technology of Cambodia in Phnom Penh, the University of Delhi in India, Mahidol University in Thailand, the Royal University of Phnom Penh in Cambodia and the Ecole Supérieure Polytechnique in Senegal. The purpose of this paper is to show the trajectory that this initiative took over its first five years and to provide a resource for other instructors and institutions working on similar long-term global software development education initiatives.

1 Introduction

Keeping software engineering education current and relevant is a challenge for university instructors. Since the term was introduced and then popularised in the 1960s [1], it could be argued that the prevailing trend in software engineering has been the introduction of a succession of new techniques and processes, which presents an obvious challenge for conducting foundational education in this area. This is illustrated, at a very high-level, by changes in the basic characterisation of software development process models over the years, ranging from the step-wise Waterfall model of the 1970s [2], the risk-confronting spiral model of the 1980s [3], the incremental and iterative models of the 1990s [4], to the agile models of the early twenty-first century [5]. Unlike the more-established fields of engineering, software engineering is a relatively young and evolving trade-oriented discipline, a factor that places some unique demands on its pedagogy. There is a need to design an educational experience that is timely and engaging for students, yet instrumental in preparing them to be productive members in both the current and future workforce.

One major trend in the software industry in recent years has been the move towards more distributed forms of working [6]. With the need to reduce development costs, increase the speed of delivery and improve quality, software products are now increasingly engineered via collaborations across people, organisations and countries. The challenges facing globally distributed software development projects of this nature have been studied and reported in the literature, generally focusing upon the economic, technical, organisational, cultural and communication issues [7–12]. Successful software development today is therefore as much about the social and team aspect as it is about the process and technology. In response, a number of university-level courses have attempted to address some of these realities, focusing on the development of software systems in a global context and on the use of collaboration tools to support it (examples being [13–18]). However, few education initiatives focus upon uniting students from across service providing and developing countries, so as to experience all sides of the offshore outsourcing relationship, and few focus upon evolving a model of working for the longer term.

In 2005, we changed the nature of the group project in the capstone software engineering course for undergraduate computer science students at Pace University in the USA to include a global teamwork dimension. Historically, the group project would bring together a team of two to four students to work on a small but realistic software development project. We changed the arrangement such that the Pace University students would collaborate with a group of students from the Institute of Technology of Cambodia (ITC) to form an extended global team of six to eight students. We established a number of projects so that the Cambodian students acted as clients for Cambodia-specific software systems and the USA students acted as developers, and we highlighted this experience in a previous publication [19].

Building upon this experience, Pace University and ITC then partnered with the University of Delhi in India in 2006 to enable the extended global team to sub-contract a well-defined component of their projects to an offshore third-party. This was to capitalise upon institution and country-specific skills, and to learn about global supply chain management in the process [20].

Rather than setting up multiple concurrent projects and getting them all partially completed, the students from these same three countries and institutions worked together on a single project in 2007, with the intention of deploying the resulting software system in Cambodia. Adapting the model of 2006, the project incorporated a competitive bidding process for undertaking an outsourced component of the work. In addition, the evolved model made use of graduate software engineering students from Pace University to help in improving quality [21]. However, this project was not itself completed to the satisfaction of the Cambodian client, so the project was repeated in 2008 to capitalise upon the artefacts developed and the knowledge gained.

In 2008, a team of Cambodian students represented the client constituency and explicitly managed the requirements for the project. To maximise the potential for acceptance of the final software system this year, five country-based teams were put into competition to develop the software system from the same set of requirements, each team being independently coached and audited by graduate students from the USA. The model involved 60 students and 7 instructors, having been extended to incorporate a development team from Mahidol University in Thailand and a team in charge of socialisation from the Royal University of Phnom Penh (RUPP) in Cambodia. The latter was specifically intended to explore the role of global team relations on quality. A software system was deployed within Cambodia as a result of this year's effort [22].

Although the various roles in the project were always distributed globally in the first four years of the education initiative (i.e. the clients were in one country and the developers were in another country), the actual development component was not distributed. In 2009, the model was therefore scaled back down to examine this new distributed working arrangement, and students from the Ecole Supérieure Polytechnique of Senegal worked with students from the USA and Indian institutions as a single set of developers. The objective was to develop and deliver a mobile application for children [23].

This paper provides an overview of the first five years of this global and long-term software engineering education initiative. Sections 2–6 explain the major annual themes of crossing continents (2005), sub-contracting to India (2006), integrating quality assurance (2007), scaling up for competition and deployment (2008) and scaling down for true distributed development (2009). Each section outlines the motivation for the year, summarises the educational set-up and reports upon the key lessons learned, leading to recommendations that influenced the theme and design of the model in the subsequent years. Although previous publications have presented details of our education initiative on a year-by-year basis, this is the first paper to bring the five years of the initiative together to highlight the trajectory that we took and to explain the rationale for its evolution. We conclude with a list of key points for other instructors and institutions that are considering embarking on a similar global software development education initiative. Finally, we draw some parallels for commercial practice.

2 Crossing continents (2005)

Full details on the first year of this education initiative are reported in the previous paper [19]. This section highlights its objectives and set-up, then focuses upon the lessons and recommendations.

2.1 Objectives

The first year of the education initiative was triggered by a noticeable decline in undergraduate computer science enrolment within a number of American universities [24], coupled by media and political spotlight on technology and software development offshore outsourcing [25]. Such messages can make it difficult for students to see both where and how they have a role to play in this new model of distributed software development. As educators, we responded to this situation by planning to educate our students about the realities of software development in this context, while preparing them for the roles that they were likely to play in a global marketplace. The key objective of this education initiative when it started was to determine the soft skills that would be required for students to work in this setting. In particular, to explore the kind of communication models that would be the most effective to elicit requirements from across continents and to ensure their satisfaction.

2.2 Educational set-up

We began by changing the nature of the group project in the capstone software engineering course taken by junior (third year) and senior (fourth year) Pace University undergraduate computer science students. Traditionally, the group project would bring together a team of two to four students to work on a small but realistic software development project for the 14 weeks of the spring semester. In 2005, we united the USA students with fourth-year undergraduate Cambodian students from The Institute of Technology of Cambodia (ITC), who were also taking a software engineering course. These students have to work together as part of an extended global team of five to seven students.

Responding to our objectives, we knew that the barriers facing global working would be pronounced in this set-up, so many issues would certainly be experienced by the students first hand. In addition to the 12-hour time difference, there were intermittent electricity, aging technology and limited Internet connectivity within Cambodia in 2005. The Cambodian students had only limited daytime access to the Internet from the ITC labs, so they relied upon the emerging cyber-cafes. E-commerce was also still a novelty to the vast majority of the population in Cambodia, including the participating students. Complicating an already challenging communication scenario was the language and cultural differences – the Cambodian students spoke Khmer, French and only a little English.

2.2.1 Students, roles and teams:

The software engineering course at Pace University comprised 19 students and the course at ITC comprised 13 students. To simulate an offshore outsourcing software development project scenario, we established the projects so that the:

- *Cambodian students were the clients and the end-users:* They owned the problem that the proposed software system was to address, knew the environment in which it was to operate and had the authority to accept the work of the developers (or not).
- *USA students were the developers:* They were to elicit the requirements for the software system from the Cambodian team members, validate them, propose design options, build the chosen design and test the eventual system, while also handling requirements changes.

Five extended global teams were set-up, each composed of three to four students from the USA acting as the developers and two to three Cambodian students acting as the clients and end-users. Note the reversal of conventional onshore/offshore roles in this scenario. We explicitly designed the model this way to give the USA students the opportunity to find out what it would be like to be on the supply side of the onshore/offshore equation and for the Cambodian students to experience the reverse. The students were free to choose their local teams and the instructors assigned the global team partners.

2.2.2 Projects: Two ITC-specific themes were proposed as candidate projects, both to automate current paper-based activities within the Cambodian school. The Cambodian students developed these themes to create their own more specific project proposals:

- *ITC schedule builder and classroom assignment system:* Two student projects were established to design and develop a software system that would generate schedules and room assignments for ITC, with respect to a set of courses to be taught in a trimester, and based upon room availabilities and professor preferences.
- *ITC student information system:* Three student projects were established to design and develop a software system that would manage student registration at ITC and provide a way to view a student's information on courses, grades, attendance records, and so on.

2.2.3 Logistics: The Pace University software engineering course spanned a 14-week period, three weeks of which were not part of the ITC calendar, so the Cambodian students used their free time to participate at the start of the project. The project milestones were organised around one week for the initialisation 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-based model with iteration and feedback cycles, mostly for instructor control and visibility, as well as for convenient synchronisation with the teaching sessions. The topics covered in the classroom included software processes, requirements, design and testing.

The local and global teams used e-mails, chats and face-to-face meetings for communications. The requirements were captured using questionnaires and chat discussions. Templates were designed by the instructors for the requirements, design and testing documents to help the students standardise their work. The developers used Eclipse, Java, MySQL and Tomcat to implement their software systems. To examine the communication models employed, the total communication activity across the extended global team (i.e. the quantity, type, participants and topics) was recorded using an online weekly questionnaire. All e-mails were archived and the chat transcripts were posted on the students' webpages. Each team also maintained a blog containing information about their activities, discoveries and difficulties, as well as their reflections about the software engineering process and their experience of global software development. This material was used to explore the issues surrounding soft skills.

2.3 Lessons learned and recommendations

The main lessons learned about the model of 2005, along with the recommendations that influenced subsequent years of the education initiative, are summarised below.

2.3.1 Lesson 1: it is easy to overlook the costs in start-up, set-up and on-going management: The instructors from the two institutions met face-to-face prior to the start of the entire education initiative to put many building blocks and ground rules for the first set of projects in place. However, in addition to the amount of prior work, the on-going control that we all found necessary to run this initiative on a daily basis was a sharp lesson about the realities of offshore outsourcing. We found that we all had to invest considerable time and had to act more like project managers than educators to keep all the student projects on track. This is very typical in offshore outsourcing projects and is frequently a 'hidden' cost that is not initially factored into sourcing decisions.

2.3.2 Recommendation 1: attend to project management in student projects as rigorously as industry projects do: Running an education initiative of this nature is not for the faint-hearted. It is going to be time-consuming for the instructors and it demands that they possess disciplined project management skills. However, students themselves could and should eventually undertake some of this project management role over time, especially once the overall process is understood and streamlined. This would also enable the students to learn about the initial and hidden costs incurred by global projects of this kind. Towards this end, the instructors need to document and review the overall process, build upon the lessons learned and seek options for delegating the non-critical aspects of the management task.

2.3.3 Lesson 2: be aware of the lag between theory and practice: When students are new to the discipline of software engineering, attempting to synchronise a practical software development project with classroom instruction about software engineering theory means that the students do not always have the knowledge that they need to look ahead in the software development lifecycle and plan their work. Consequently, they learn about requirements when they need to do requirements and they learn about testing when it comes to testing. This means that there is rarely sufficient time built into the project timeline for students to reflect upon feedback, improve practical skills and to re-do work. Students therefore do what they need to do to get the job done and to get the grade, understandably, usually with little enduring consequences at project completion if the software system is not deployed. Moreover, the software engineering good practices that the students are expected to apply require some depth of understanding and take time to learn, so students may form negative impressions of the utility of the practices under study if they are always having to find an expedient work-around (and getting away with it).

2.3.4 Recommendation 2: account for just-in-time learning and capitalise upon existing expertise: If participation in a global software development project is to run concurrently with a first software engineering course, then there is a need to be less demanding in the breadth of the software engineering skills that the students need to acquire and apply if they are to be successful overall on a project of this nature. This could be facilitated by tightening the learning objectives of the courses of the various student constituents and by allocating well-delineated project roles across countries based upon these objectives; otherwise, student constituents with specific background and expertise could be incorporated into the model to tackle well-defined components of the overall project, relieving some of the in-depth technical work required from other students.

2.3.5 Lesson 3: maintaining objectivity is difficult, but essential: It is natural that the participating instructors wanted their students to succeed and for the global software development project to be a success. We therefore found that it was important to have an independent faculty member keep sight of the bigger picture on the overall project, looking across the individual projects for commonalities, differences and problem areas, while also monitoring the quality of the processes and evolving software products more objectively. This reflects the critical role of governance in offshore-outsourced projects.

2.3.6 Recommendation 3: establish independent third-party oversight: Assign an independent third-party for the governance role on any global software development education initiative. This may be an additional faculty member from one of the institutions or, better, someone with no connection to the participating stakeholders. However, there is equally an important need to educate students about the oversight and governance that is required to support global working of this kind. To maintain the independent perspective that is demanded, senior and more experienced graduate software engineering students could potentially be integrated into the model to assume some of this governance role and to learn some of its practices.

2.3.7 Lesson 4: do not assume that soft skills come easy: Despite our best intentions, we focused more upon the technical skills in the classroom lessons and left the soft skill acquisition somewhat to chance. The students experienced issues associated with setting up shared meeting times, organising themselves, making assumptions about the clients and their requirements, dealing with changing requirements and sharing information among an extended global team. We obviously need to prepare the students for and assist them more with these soft skills. In studying the communication models that emerged from within the various teams, we found that those teams that used a single point of contact for communications with the offshore team members experienced fewer coordination problems overall, so this may be an obvious place to start with the softer skills focus.

2.3.8 Recommendation 4: create roles and simple protocols to nurture soft skills: A balanced emphasis on technical and soft skills is important to the overall success of global software development projects, but finding a way to achieve this balance in an engineering curriculum can face some resistance by instructors and students alike. Appointing a communications leader per local team, and providing simple guidelines on the communication expectations and protocols, may be just the seed that is needed to make the students feel more comfortable to communicate and work together.

3 Sub-contracting to India (2006)

Full details on the second year of this education initiative are reported in a previous paper [20]. This section highlights its objectives and set-up, explains how it accommodates the recommendations from the previous year, then focuses upon the new lessons and recommendations.

3.1 Objectives

Increasingly, software houses outsource well-defined components of their contracts to countries where there are the requisite skills and/or the potential to sustain continuous effort around the clock. In an attempt to relieve some of the technical burden on students (recommendation 2), and to focus on nurturing both the managerial and soft skills (recommendations 1 and 4), we explored the role of sub-contracting in year two of the education initiative. Setting up a supply chain model of software development would require students to learn how to divide up a project into component parts for different parties to work on which, in turn, would emphasise the role of a shared and systematic process with critical liaison points. In addition, there would be more focus upon scoping and delineating work boundaries, eliciting requirements from remote clients, communicating an understanding of the requirements back to the clients and then onto third-parties, and learning about the testing and integration that is needed to assemble a complete software product. We were particularly interested in exploring the social activities that would foster a ‘whole’ team concept in such a distributed setting. Note that we did not intentionally set out to reduce the work of the instructors this year; our process was far from streamlined and we were still learning. Also, while an independent faculty member oversaw the projects this year, the role was quite a passive one, so recommendation 3 was only partially addressed at this time.

3.2 Educational set-up

In 2006, the model from the first year of the education initiative was extended to include first-year graduate computer science students from the University of Delhi in India. These students were concurrently taking a course on database system implementation, so they had expertise in database design and could relieve some of the technical demands on the undergraduates who had yet to learn this topic. Although the Cambodian students remained as clients, the provisioning of the solution was changed. The Pace University students were to sub-contract part of the software system design and development to the students from India, while also managing the end-to-end contract.

The model was established such that there were no contacts between the students in India and Cambodia, as initially the Indian students were intended to act as pure suppliers to the USA developers. The intention was to give the Indian students a realistic experience of working with pre-specified requirements and on learning how to work as part of a supply chain. The socialisation focus was upon nurturing the client/developer (Cambodia/USA) relations, but not so much upon the developer/sub-contractor (USA/India) relations, assuming that the latter would follow naturally. What was surprising was that the Indian students decided to develop their own version of the software system from the requirements specification, of their own volition and in their own time. At the end of the semester, the Cambodian students therefore assessed the software system developed by the USA students (with Indian sub-contracting) and compared this with the version developed solely by the Indian students.

3.2.1 Students, roles and teams: This year of the initiative involved 11 students from Pace University, 16 students from ITC and 6 students from the University of

Delhi. It consisted of three extended global teams, each composed of 10 to 12 students distributed among the three locations, three to four students from the USA acting as prime contractors, five to six Cambodian students acting as clients and two Indian students acting as sub-contractors. The term 'local team' was used to refer to co-located team members. The students were free to choose their own local teams, which each had their own assigned team leader, communications leader and quality assurance leader. The global team partners were also student selected. With respect to the extended global teams:

- *Cambodian students were clients and acceptance testers:* Their responsibilities were to describe the software they wanted to be built and the context in which it was to operate. They also had to review and give feedback on the requirements, design and testing documents, test the software and submit bug reports. Their responsibilities also included reporting on the experience of working with the USA students.
- *USA students were developers and lead contactors:* Their responsibilities were to elicit the requirements from the clients and produce an agreed and validated requirements specification, propose design options that sub-contract part of the software system design and development to the Indian students, implement the software and test it, while concurrently handling requirements changes, integrating feedback and managing the end-to-end contract. They were to report on the experience of working with the Cambodian and Indian students.
- *Indian students were third-party suppliers:* Their responsibilities were to provide the USA students with a database design and the corresponding structured query language (SQL) code to be integrated into the overall software system design. Their responsibilities also included reporting on the experience of working with the USA students.

3.2.2 Projects: The instructors proposed three projects that were each specific to the Cambodian context:

- *ITC library management system:* To design and develop a software system to replace the mainly paper-based activities of the ITC central library.
- *Cambodian crafts on-line store:* To design and develop a software system that would sell Cambodian crafts via the Internet.
- *Cambodian on-line restaurant:* To design and develop a virtual restaurant selling Cambodian Khmer dishes for home delivery via the Internet.

3.2.3 Logistics: The project timelines were aligned with the academic calendar of the students at Pace University, even though there was overlap with the calendar of the Cambodian and Indian courses. The students followed a lightweight waterfall-based process model, with feedback and iteration once again, for fixing milestones and aligning schedules. It included one week for initialisation of the project, five weeks for requirements, three weeks for design, three weeks for coding/construction and two weeks for testing. The design phase included a faculty-managed request for proposal (RFP) for the database component of the three web-based projects. One of the participating USA instructors oversaw the three locations and played the role of project manager. This was particularly important to assist with this first attempt of including a supply-chain scenario in the initiative, as it incurs critical coordination and integration logistics that need to be understood.

The developers used Eclipse, Java, MySQL and Tomcat to implement their software systems. The concurrent versioning system was introduced for code sharing and versioning management. The teams shared their work by posting documents on websites and maintained blogs about their experience on the project. Asynchronous communications were orchestrated via mailing list and synchronous communications took place via chats.

3.3 Lessons learned and recommendations

The main lessons learned about the model of 2006, along with the recommendations that influenced subsequent years of the education initiative, are summarised below.

3.3.1 Lesson 5: assumptions about team members materialise from day one: As per the first year of the education initiative (lesson 4), the lack of social connection between the students across the globe was found to be a major impediment to smooth collaboration. This was particularly apparent at the onset of the project when relationships are formed and trust is established. While the instructors carefully planned for the relationship between the USA and Cambodian students, the USA and Indian relationship was left to chance. Consequently, photos of global team members in India were not distributed in a timely fashion and incorrect first impressions that were made in the USA were not easy to rectify later. Social bonding, such as getting to know all the team members and their wider interests, lies at the heart of international relationship management and has repercussions for the health of a global supply chain.

3.3.2 Recommendation 5: focus more than you expect upon social bonding activities and communication protocols, and from day one: There is a need to instigate a socialisation process at the onset of a global software development project for all stakeholders and to attend to this socialisation periodically throughout the project. Students need to learn more about the backgrounds and cultures of each other, as this sets up expectations and protocols for communication across social boundaries. It is also likely that this has ramifications for the productivity of the extended global team and the quality of the work that they can produce together. Undertaking some socialisation activities that enable students to get to know each other, not only as developers but also more broadly as individuals, could achieve this objective somewhat surreptitiously.

3.3.3 Lesson 6: if you do it for them, then they do not learn: Even though it was our first lesson (lesson 1), we acted even more like project managers than educators in this second year of the education initiative and encountered even more unexpected work. This was due to the added complexity of integrating a new country partner and planning how to incorporate sub-contracting into an already challenging model. It became difficult to align three sets of academic calendars and demanded even more time from the instructors to coordinate the disparate work, ensure that the projects completed in a timely fashion and ensure that all the students achieved their learning objectives. Such a situation where the instructors take over too much of the running of a project can prevent the students themselves from experiencing important milestone setting, planning and coordination tasks, and so committing to them. There is a need to balance more of this task with students so that they understand its role in running a successful project and buy into the decisions that are made, as well as to make the situation viable for the instructors.

3.3.4 Recommendation 6: establish support systems for both students and instructors: It can be difficult for instructors who want their students to succeed to relinquish control on a project and to leave critical tasks that impact the entire global software development project to students. However, a balance needs to be reached. To prevent overwhelming already over-tasked students who are struggling to learn how to be effective clients and developers, professionals and/or more experienced software engineering students could potentially assist the instructors by working with the students in a mentoring or coaching capacity to achieve some of these project tasks. As a first step, they could provide support to help the students plan and coordinate their work so that it aligns better with that of their extended team members.

3.3.5 Lesson 7: competition is not always perceived as fair: The Indian students wanted to create their own version of the software system, in competition to the USA students, and proceeded to do so in their own time. The ensuing competition between the two sets of developers in the USA and India hence impacted what was meant to be a useful sub-contracting relationship, particularly in terms of trust and communications. Moreover, the USA students were aware that they were not competing fairly as they had to undertake substantially more project responsibilities than their emerging Indian competitors, including liaison with the Cambodian clients to prepare the requirements specifications that the Indian students then also worked to.

3.3.6 Recommendation 7: if you want healthy competition to improve quality, plan for it: This competition did encourage the pursuit of quality by both the teams and, used judiciously, competition could increase the likelihood of delivering high-quality and deployable software systems. While students can benefit from some healthy competition, this needs to be carefully factored into an education initiative though, rather than left as an emergent and unplanned element. In our case there was an uneven distribution of workload, so it was not perceived as fair to compare the two resulting software systems and this was a source of contention.

3.3.7 Lesson 8: do not try to run disparate projects with the latest technologies until the underlying process works: With any supply chain, transparency in communication is desirable, especially so when students are learning about subtle inter-dependencies and trust. Coupled with this is the important role of having a shared process to integrate into and an up-to-date version controlled repository for information exchange. There were a number of problems associated with artefact versioning and gaining access to up to date information this year. We provided students access to the latest tools and technologies to use for all the aspects of the intended software process, but we did not pay sufficient attention to how the students could exploit these tools with straightforward communication and change management processes in the classroom sessions.

3.3.8 Recommendation 8: focus upon the process before the project topics and tools: Do not start off too ambitious by running multiple projects on diverse topics and attempting to use the latest technologies before you have the processes and protocols for running the projects in place and streamlined. More attention needs to be paid to designing simple processes for software development tasks and simple protocols for communications, in advance of introducing students to any technologies chosen to support them. While it does not address all of these issues, the introduction of an agile process and its end-to-end tooling could permit the students to have a ready-made framework for communications and also reduce the number of tools that the students would be required to learn.

4 Integrating quality assurance (2007)

Full details on the third year of this education initiative are reported in the previous paper [21]. This section highlights its objectives and set-up, explains how it accommodates the recommendations from the previous year, then focuses upon the new lessons and recommendations.

4.1 Objectives

To address the need to focus primarily upon the process rather than on fragmented project topics (recommendation 8), year three of the education initiative was designed to provide a single project that would require the students to share a goal and take the software system they developed through to production quality. The students were to learn about ‘through-life’ software development processes and the ‘total cost of ownership’ of software, and the instructors were to focus upon streamlining the end-to-end process. Building upon the experiences of 2006, integration was to be a particular focus.

Pursuing the sub-contracting theme of 2006 (recommendation 2), and now planning for an element of competition (recommendation 7), a student-managed RFP process was to be conducted for a well-defined sub-component of the project work. An associated research aspect here was to examine the impact of the students preparing, responding to and evaluating an RFP on the quality of the written requirements and the eventual design resulting from it. A further objective was to investigate how to incorporate graduate software engineering students into the model so as to relieve the instructors of some of the day-to-day work spent on attempting to manage and improve the quality of the students’ work (recommendations 1 and 3), to coach the students on particular software engineering techniques and integration matters (recommendation 6), and to assist with soft skill acquisition (recommendation 4). The aim was thus to explore models through which graduate and undergraduate students could work together, with pedagogical value to both sides. To facilitate all this change in the model, more attention was to be placed on social bonding activities both before and during the project (recommendation 5).

4.2 Educational set-up

In 2007, we focused upon a single project that was important to ITC and which we intended to deploy in Cambodia. The students worked in sub-teams on separate components of a larger project that would demand full-team integration. We expanded the model to include students from the

Master's Program in software design and engineering at Pace University to help with mentoring, auditing and software quality assurance.

4.2.1 Students, roles and teams: The undergraduate course at Pace University comprised 8 students, the course at ITC comprised 13 students and 6 students were drawn from the University of Delhi course. Seven Pace University graduate students also participated. The Cambodian students were again the clients, but this time the students were split into sub-teams to deal with the requirements of a sub-component of the overall software system. The USA undergraduates were also split into development sub-teams to work directly for the client sub-teams. The sub-teams would then need to integrate to give a complete system. The Indian students were organised as three teams of two students. Each pair was to bid for the database work of the overall system. The winning database design was then to be developed by the full set of Indian students. In addition, the Cambodian students were to prototype their own version of the overall software system to satisfy their local course requirements.

One graduate student was assigned to each USA undergraduate sub-team to act as its mentor. A third graduate student was selected as a floating mentor for all the sub-teams to assist the students with planning and undertaking the integration aspects of the project. The three mentors were working in the software industry full-time while taking their graduate degree. The remaining four graduate students acted as auditors. They were organised into pairs and assigned to the two USA sub-teams to monitor and report on their progress. They produced audit checklists and organised audit interviews of the developers to provide process feedback.

4.2.2 Project: The objective of the project, called MultiLIB, was to develop a multi-purpose web-based library management system for the internal library of the Department of Computer Science at ITC. The development effort was partitioned into sub-teams in the following way:

- *Librarian side:* To focus upon the part of MultiLIB to be used by the Librarian for managing all the resources, the accounts in the system and the borrowing/return transactions.
- *Student side:* To focus upon the part of MultiLIB to be used by the students to view the available resources in the library, to reserve the resources and to consult the status of personal accounts.
- *Innovation side:* To focus upon the part of MultiLIB to be used by the students to view the electronic resources, such as e-books, audio and video. This was a client-only role and the sub-team was to be forward looking to consider future innovative features likely to be required of MultiLIB in later versions.

4.2.3 Logistics: The single extended global team followed a loose waterfall-based development process with iteration based upon feedback from the clients, mentors, auditors and instructors. The project spanned 14 weeks comprising two weeks for set-up and team bonding, concurrent with six weeks for requirements, four weeks for design and four weeks for coding and testing. The tooling converged to Eclipse as the development platform, with JUnit for testing, Subversion for version control and java.net for bug tracking. The communication tooling comprised chats and separate mailing lists for each side of MultiLIB and for each RFP. The mailing lists were combined during integration. Websites were replaced by wikis to store and disseminate project artefacts, which thus served as the coordination backbone of the overall project; the wikis contained all the artefacts that were produced by the teams and permitted the students to increase their mutual awareness of project status.

4.3 Lessons learned and recommendations

The main lessons learned about the model of 2007, along with the recommendations that influenced subsequent years of the education initiative, are summarised below.

4.3.1 Lesson 9: students experience tunnel vision: Yet again, the Indian students decided to develop their own variant of the final software system, entirely unplanned for and of their own volition. However, they did this based upon a superseded version of the requirements specification. While the new version of the requirements was posted on the global project wiki, the Indian students were so busy with their own development work that they did not think to look for updates, not expecting the requirements to change so late on in the process.

4.3.2 Recommendation 9: push critical project information to students: It can be problematic to rely upon a pull-type information dissemination model when work is under way and when deadlines are pressing. Critical requirements changes and project updates need to be pushed or broadcast to team members or to selected representatives on a project to disseminate the updates as necessary. Versioning also needs to be anticipated by students, so think about having a change 'drop-dead-date' for changes to assist them, and change management practices factored into the process guidelines explicitly.

4.3.3 Lesson 10: friends commit to each other: Considerable energy was expended at the beginning of the project to create a bond between the USA and Cambodian students through the exchange of gifts, personalised postcards and videos. The consequence was that the Cambodian students were more committed and motivated to work on the global version of the project as clients for the USA students than to focus upon developing their own local version of the software system as required for their course. In fact, they did not complete the local version of the software system and lost motivation for it. Additional factors here were the already high workload and the non-synchronisation within the taught course of the technology skills that were needed for the full implementation of the software system (lesson 2 repeating once again). There is an on-going tension between running a successful global and collaborative project and accommodating the wider curriculum demands of each participating institution and course, while simultaneously expecting all the students from across the globe to have a similar project experience and to come out with all the required global software development skills.

4.3.4 Recommendation 10: establish 'official' international community relationship managers: We recognise that there is considerable value in socialisation activities, but running and undertaking these activities takes time and can distract from what instructors perceive to be the 'real' work on a project and core learning objectives. Some students should perhaps act as international community relationship managers, charged with creating and running regular activities to ensure that all the students learn more about each other and work well together. The demands and value of this role need to be appreciated by all the instructors for this to happen. Alternatively, a third-party that is not actively engaged in the technical aspects of the project (so perhaps with different course demands) could potentially fulfil this important role.

4.3.5 Lesson 11: mentoring or coaching is the next best thing to apprenticeship: The graduate students who operated as mentors this year performed a role that was much appreciated by the undergraduates and relieved some of the additional work previously undertaken by the instructors. In their final report, the mentors stated that they felt more like coaches than mentors, reflecting the fact that the assistance they gave was directed more towards a specific project and its tasks rather than towards longer-term career advice. Engaging senior graduate students to guide the trainee software engineers reflects the apprenticeship model that is so widely used and successful in industry.

4.3.6 Recommendation 11: find a way to engage external students as the instructors' partners: Providing mentors and/or coaches is an ideal way to take some of the pressure off instructors. Using more experienced students may also provide a 'safer' sounding board for undergraduate students who may not want to reveal any misunderstandings or vulnerabilities to their instructors. For this support to have a positive impact on the work of student teams, specific coaching tasks need to be planned for ahead of time from within an understanding of the global project needs and syllabi topics, and built into the project timeline directly.

4.3.7 Lesson 12: effective auditing demands cycle time: The auditing role was not as widely appreciated by the students as we anticipated. This was mostly because the audits were not conducted in a timely manner. It took too long for the audit checklists to be prepared and then for the interviews of the students to be carried out by the auditors. The instructors and mentors also did not receive the resulting audit reports until a time by which some of the feedback was redundant to pass onto the undergraduate students. However, the audit process did catch some issues early on in the project, issues to do with conflicting team perceptions as to the responsibilities for components of the project, and it made recommendations for minor but critical improvements to the requirements specification document.

4.3.8 Recommendation 12: factor in time to act upon audits: Audit plans and checklists need to be in place early on in a project, alongside the other project plans, and a timely feedback loop needs to be built into the project timeline for students to incorporate the results of any auditing activity. Students must be given the opportunity to learn and improve their processes for auditing effort to be useful and valued.

5 Scaling up for competition and deployment (2008)

Full details on the fourth year of this education initiative are reported in the previous paper [22]. This section highlights its objectives and set-up, explains how it accommodates the recommendations from the previous year, then focuses upon the new lessons and recommendations. More details on the infrastructure and socialisation aspects of this particularly challenging year are provided in [26, 27], respectively.

5.1 Objectives

The 2007 MultiLIB software system did not get deployed in Cambodia, as the client was not fully satisfied with its quality. The extended global team faced many challenges with respect to changing requirements, product integration and cross-role communications. In 2008, we therefore aimed to capitalise upon the work achieved in 2007, with the main objective of increasing the likelihood of getting the new software system accepted by the client and then deployed into operation in Cambodia.

Consequently, five development teams were put into competition to develop five versions of the software system from the same set of requirements to maximise this likelihood (recommendation 7). The requirements specification was therefore a core focus of this year and it was permitted to change up until a couple of weeks before the delivery was due, so the project timeline was clear and requisite procedures were put in place to push the updated information to facilitate any necessary change (recommendation 9). The positive experience of incorporating the mentors and the promise of auditing encouraged us to persist with a model that included these two additional and external roles to assist with and assess the work of all the participating students (recommendations 11 and 12).

In addition to addressing all the recommendations from the previous years (recommendations 1 through 8), the 2008 setting was specifically intended to study whether taking the effort to get to know the other team members from across the globe actually had an impact in terms of more effective communication and whether this, in turn, resulted in higher quality work. We therefore ran a controlled experiment with an additional set of students to focus upon international community relations (recommendation 10).

5.2 Educational set-up

The 2008 model was extended to include undergraduate students from Mahidol University in Thailand as developers (one of the competing development teams) and undergraduate students from the Royal University of Phnom Penh (RUPP) in Cambodia (in the socialisation leader/international community relations role). Four countries, namely the USA, Cambodia, India and Thailand, and five institutions, namely Pace University, ITC, the University of Delhi, Mahidol University and RUPP, were therefore involved in the project this year. The mentor–auditor model was improved upon by providing coaches to both the clients and developers, and auditors were assigned to each competing development team to assess the quality of their process and resulting software system, and to provide continuous feedback to encourage process improvement.

5.2.1 Students, roles and teams: Overall, 60 students organised into five competing development teams participated in the project in 2008 (two teams in the USA, but at different campus sites, one team in Cambodia, one team in India and one team in Thailand). The extended global teams were composed of clients, developers, quality coaches and quality auditors:

- *Cambodian students were clients:* Five students owned and managed the requirements. They were also responsible for assessing the quality of the software systems produced by the development teams and for selecting the highest quality software product for deployment.
- *Development teams were from four different countries:* Five teams composed of four to six students worked to deliver a software system that satisfied the provided requirements. Each team was sponsored by a quality coach and supported by a team of quality auditors. The students were either following or had previously taken a software engineering course, except for the Indian students who were graduates enrolled in a database course (as in the past years).
- *USA graduate students were quality coaches:* Five students helped the client team to baseline the requirements specification, manage changes and select the final software system to deploy. Five additional students mentored each of the five development teams and ensured regular communications between the clients and developers.
- *USA graduate students were quality auditors:* Fifteen students, also employed in a technical capacity in industry full-time, acted as auditors. They designed audit checklists and proposed recommendations to improve the developers' processes and work products. One student assumed the overall quality management role and thus assisted the instructors. The auditors, as well as the coaches, were registered in Pace University's software engineering and design graduate program.
- *Undergraduate students in Cambodia were socialisation leaders:* Two students from RUPP were tasked to introduce one of the USA teams to Cambodian life and culture. This was undertaken to study the impact of dedicated socialisation in global software development projects by cross-comparing the experiences and results of the two USA development teams.

5.2.2 Project: The 2008 project was the same as the 2007 project, MultiLIB, but augmented by additional functional and non-functional requirements.

5.2.3 Logistics: Considering the number of students and institutions involved in 2008, the planning of the project involved considerable prior discussion between the instructors. The teams used a common software development process and timeline to synchronise the effort across locations. This was a loose waterfall-based process with feedback cycles. The project spanned 19 weeks, with four weeks for requirements overlapping with two weeks for set-up, four weeks for prototyping and design, six weeks for coding and testing and five weeks for deployment. The quality of the final software system was determined by the clients, with the assistance of the client coaches, based upon the number of requirements satisfactorily implemented and their relative importance.

The development teams were free to employ the technologies they had available at their own local sites for development. Their use of technology converged to Eclipse or Netbeans, JUnit, Subversion, MySQL, Tomcat and java.net, except in Thailand where Visual Studio was used to develop a web-based application in C#. Each team was supported by a mailing list and a wiki to organise the artefacts they produced, creating a single repository for each development team. The teams were provided with tutorials and videos to learn to use some of the shared tools.

The 2008 set-up also integrated the use of specific socialisation tools and exercises to study the impact of socialisation on global software development projects. A second life island (http://www.secondlife.com) was created with entertainment areas and landmarks from different countries involved in the project. The RUPP students and one of the USA teams met weekly in this virtual environment to discuss the aspects of Cambodian life and culture. Given that the clients and end-users for the intended software system were Cambodian, this was directed at improving and smoothing relations with these stakeholders for this particular development team. More broadly, fun socialisation exercises were designed and administered for all students at key points during the overall project timeline to determine how much the members of the extended global teams knew about each other. The first exercise examined whether students were aware of the map location, famous monuments and culinary dishes of the other countries they were collaborating with. The second exercise examined whether the students could identify their global team partners out of the pictures of the faces of the 60 participants. The results of these experiments are discussed elsewhere [27], but suggested that socialisation activities pay off and are a valuable investment of time in global software development projects; come the close of the project, the teams that knew each other better produced the highest quality software systems.

5.3 Lessons learned and recommendations

The main lessons learned about the model of 2008, along with the recommendations that influenced subsequent years of the education initiative, are summarised below.

5.3.1 Lesson 13: you can scale up if you have the basic model in place: Undertaking global software development projects of this scale with students requires thorough planning and substantial preparation before they start. During the course of the project itself, the instructors needed to monitor progress and be on call at all times as the aligned deadlines had to be met by the various students for the overall project to succeed. There was little room for flexibility and problems. Having a working model in place so that the instructors all knew what needed to be done, when and by whom, along with an awareness of the likely time commitment, meant that we could scale up and examine some new research questions this year. This is not to say that it was easy though.

5.3.2 Recommendation 13: if you change too many things, scale the model back down: There is a need to streamline the overall process of an education initiative like this in the small before taking on a more ambitious project like we did in 2008; it is easier to scale with familiarisation of all the activities, issues and mitigating solutions related to running a small project. However, if you change too many things in your model, such as the development process, the technologies and the countries involved, it may be prudent to scale back down again initially and to re-integrate practices only gradually.

5.3.3 Lesson 14: greater awareness of extended team members matters: The students in our 2008 setting came from different socio-economic and cultural backgrounds, and so the interactions with extended team members did not always proceed as students expected. The socialisation activities that we conducted helped the students to develop a greater awareness of each other's contexts and thus led to an enhanced understanding of the different social protocols to use for communicating. When the students began to know each other better, they felt more comfortable communicating, and project questions got posed and answered more expeditiously. This appeared to correlate with the improved quality of the process and of the resulting software systems they produced.

5.3.4 Recommendation 14: run periodic exercises or games so that students get to know each other: Socialisation exercises are great tension breakers for students and they also add an element of fun to what is a demanding piece of work for all the students. They also appear to add value in terms of eventual quality. We strongly recommend that other instructors begin to experiment with exercises along the lines of our 2008 maps and faces as an important adjunct to the technical work (see [27]).

5.3.5 Lesson 15: to win the quality war you may lose some battles: Our experience with the 2008 project illustrated that some students often performed to the best of their ability when they were competing with other teams. In the highly functional development teams of our setting, the competition fostered challenge and much enthusiasm. However, it was a source of stress in some of the other development teams, especially where the acceptance of the competition's demands varied among internal team members. While putting five student teams in competition led to excellent high quality work and a deployed software system, the positive experience was not shared by all. This situation obviously needs to be attended to.

5.3.6 Recommendation 15: monitor the 'health' of all team members regularly: While competition is recommended as a motivator for some students, it is important to also consider other forms of motivation so as not to alienate other students. It is equally important to focus upon the intra-team relations, because competition can be a significant source of tension when personal motivation differs among team members. We found it useful to conduct weekly (simulated) 'blood pressure' surveys, but any mitigation of the emerging problems did not happen soon enough. Daily progress meetings would be one way to promote fuller team awareness and ensure that problems get both aired and addressed.

5.3.7 Lesson 16: it is not sufficient to assign a real-life project; it is also important to make it deployable and sustainable: The students participated in a real-life project that ultimately got deployed in Cambodia. The deployment process was enabled by ensuring that there were no incompatibilities with respect to the operating systems of the students' work, by ensuring that the code was commented and easily modifiable, and by attending to projected localisation problems caused by making assumptions about the end-user context (e.g. the policies for loaning books from a library in Cambodia are not the same as those in the USA). While the competing software systems had satisfied the functional requirements quite similarly, it was the attention that the students had paid to satisfying the non-functional criteria that became defining in the final software system selection. Furthermore, having a champion in Cambodia was a key ingredient to a successful deployment process and essential for long-term sustainability.

5.3.8 Recommendation 16: attend to non-functional requirements and secure in-country champions for ease of deployment: Care needs to be taken while preparing the non-functional requirements on a student project as these have the greatest impact on the ease of deployment of any delivered software system. These are the requirements that the students found the most difficult to express, communicate and measure in our education initiative, and there is a need to attend to them more carefully in any associated courses and coaching. Having a project champion on-board can smooth over any initial problems with the deployment process and is integral to ensuring the life of the software system after the project ends.

6 Scaling down to explore true distributed development (2009)

Full details on the fifth year of this education initiative are reported in the previous paper [23]. This section highlights its objectives and set-up, explains how it accommodates the recommendations from the previous year, then focuses upon the new lessons and recommendations.

6.1 Objectives

During the prior four years of the education initiative, while the clients and developers were distributed and had to work together to elicit and validate requirements, the actual development work was not distributed. Having run this model for four years and scaled up successfully in 2008, we now decided that it was time to explore this more challenging distribution scenario in 2009. This would place higher demands on the frequency of the communication needed across countries and present some significant synchronisation issues. We distributed the development work across three locations in the USA, India and Senegal. Given the additional challenge of integrating a new partner institution from Africa (Ecole Supérieure Polytechnique in Senegal), we therefore decided it was prudent to scale back down (recommendation 13), and to eliminate some of the elements like competition, socialisation and auditing (recommendations 7, 10, 11, 12 and 14) until we had the basic model in place again. Since all the students were undertaking this work in their own time, we selected students with prior software engineering course experience and we provided them with advance materials to learn about the new processes and technologies they were to employ (recommendation 2).

The waterfall-based process model that we had been using in the previous years had some drawbacks. For instance, the time scheduled for testing was always insufficient for the teams and the developers were not required to make demonstrations to the client until the end of the project, so they rarely benefitted from detailed client feedback. We therefore decided to adopt a more agile development process [5], complemented by the Scrum project management framework [28]. Together they emphasise constant and active client/developer communication and the delivery of working software at regular junctures. They afford a number of built-in 'ceremonies' to promote team awareness of progress (recommendation 15), communications (recommendations 4 and 5) and project management (recommendation 1). They also advocate roles to promote process coaching and product champions (recommendations 3, 6 and 16). Finally, we reduced the tooling to one environment to ease the students' burden of having to learn multiple tools, each supporting separate functions, giving an end-to-end tooling solution that is firmly grounded in the targeted process (recommendations 8 and 9).

6.2 Educational set-up

In 2009, we concentrated on only one project, a mobile application for first-grade children. The project involved a single team of developers distributed across the USA, India and Senegal. Unlike previous years, the project was not integrated within a curriculum course and all the students volunteered to participate on top of their other school commitments. This made for a small team of students and it was problematic to integrate additional sets of students in supporting roles.

6.2.1 Students, roles and teams: Five graduate students were involved in the project: one from the USA, two from India and two from Senegal. All the students were developers and, for each of the three sprints (the agile term for short two to three-week development iterations), one student from each country was assigned to the role of Scrum Master, the role that guides the team through the development process and helps the team to resolve any issues. An instructor, a certified Scrum Master, played the role of product owner (i.e. client and project champion) and provided the vision and the requirements for the project. A professional Scrum Master external to the university setting played the role of the process coach. Together, these two roles provided objective feedback to guide the extended global team and to assist the team's Scrum Master on a weekly basis.

6.2.2 Project: The product owner proposed the development of target first grade, a bilingual (French and English) educational Java ME mobile phone application for children of five to six years old. The application would target children living in developing countries and was to be used by pupils of large classes supervised by schoolteachers. The application was to contain multiple-choice questions on mathematics, reading, writing and geography, and was intended to be used to support practice or testing. The feedback and scores were to be sent to parents and teachers via short message service (SMS).

6.2.3 Logistics: The 2009 project ran for nine weeks with the first three weeks devoted to training, where students were provided with documentation and video tutorials. The following six weeks were divided into three two-week sprints. The product owner provided the developers with a backlog of user stories (i.e. a collection of requirements to be implemented). At the start of each sprint, online chats of two to three hours took place to plan which requirements would get implemented and by whom. Daily scrums, effectively a short report on the progress, intentions and issues of each global team member, were posted online. One weekly chat accommodated a synchronous scrum for the global team. At the end of each sprint, the developers were asked to prepare a video product demonstration for the product owner, and to conduct a retrospective to discuss the lessons learned and the intended improvements for the next sprint.

The extended global team used IBM rational team concert (RTC) to support the development of their software system. RTC is built upon an environment that all the students were familiar with (Eclipse) and provided capabilities for agile planning, source code management, user stories, bug tracking management and reporting, along with wikis for planning and sharing artefacts, and an alert mechanism to promote team awareness.

6.3 Lessons learned and recommendations

The main lessons learned about the model of 2009, along with the recommendations that serve to take the education initiative forwards into the future, are summarised below.

6.3.1 Lesson 17: active training is more effective than passive tutorials and videos: The students were assigned readings, tutorials and videos on Scrum, agile, RTC and Java ME and were expected to undertake this preliminary work during the first three weeks of the project. However, the students did not undertake this preliminary work as conscientiously as they should have, partially due to the fact that they did not have to provide the instructors with any homework based on this work and they were not assessed on the knowledge they gained. The students mostly waited until the beginning of the project proper to learn by doing, and as needed, returning to the issues associated with having to undertake the practice alongside the first exposure to the theory (lesson 2).

6.3.2 Recommendation 17: always facilitate any preparatory work expected of students: There is a need to consolidate a student's understanding of the process, and their familiarity with the tools to be used to support the process, before the application of the skills is needed. If the students are undertaking the project of their own volition and to further their personal studies, instructors also need to be prepared to work outside the scope of their primary teaching responsibilities to support their preparations. It is important to facilitate small exercises to encourage and help the students to master new techniques, processes and tools, and not simply assume they will be undertaken and understood. The ideal solution would be to catalogue the skills required well in advance so as to ensure that they are taught and practised at earlier points in the overall curriculum. Taking such a long-term and holistic view of the computer science curriculum presents its own set of challenges.

6.3.3 Lesson 18: progress meetings need to be done regularly to be effective: For agile projects, daily scrums are important ceremonies that create transparency of work and ensure for developers' accountability. They are usually held synchronously when co-located, but in our global setting they need to be held asynchronously and so they took the form of written daily logs. These logs reported upon what each student did today, plans to do tomorrow and any impediments to their working. However, the students were not consistent in recording this information on a daily basis, so it was not possible to distinguish between those days when the students worked on the project and those days when the students were simply too busy to update their daily log. In addition, there was often an inconsistency between what the student developers said they were going to do and what they actually did, so it was frequently difficult for the other students to align. This also reduced the opportunities for the process coach to track the students' work and to give them timely feedback. Further, because the team members did not record their general availability to work on the project upfront, this caused confusion in the project timeline and led to many synchronisation problems (i.e. unlike an industrial project where employees work every day from 9 am to 5 pm consistently, e.g. volunteering students may take random time off).

6.3.4 Recommendation 18: use simple calendar, scheduling and notification tools to assist meeting logistics: There is a need for collaborating students across the globe to share their personal and academic schedules at the beginning of a project when they are sharing a role. They then need to commit to their planned work times and work tasks. Explicit work dependencies are to be expected when a development role is shared, and there is a need to undertake daily progress meetings and artefact updates so that team partners across the globe can exchange and continue with work. Tooling could and should facilitate this team awareness, meeting scheduling and dependency management more actively. Automatic reminders can be easily configured and an analysis undertaken of the impact of any availability changes or late completed work on the overall schedule.

6.3.5 Lesson 19: tasks need to be of a granularity that promotes estimation and completion: The students decomposed and estimated the tasks to undertake in each sprint based upon the user stories provided by the product owner. However, the tasks were generally too coarse to be estimated accurately by the students and their estimates as to what could be achieved by the extended global team were often unrealistic, certainly impeded by the observation above. As a result, the tasks were often not completed as planned and often carried over from one sprint to the next. The selection and estimation process improved little from sprint to sprint.

6.3.6 Recommendation 19: review planned tasks and help students improve their global estimating skills: Students need to learn to break down work into finer-grained tasks that can be readily estimated and completed as planned. There are new skills associated with agile working, such as these estimating skills, which need to be carefully attended to by the instructors, particularly since the students face the additional complexity of factoring in shared work effort that is intended to be undertaken by various team members from around the globe. All committed tasks and their estimates should be reviewed with the product owner and process coach. At the end of every sprint, a careful analysis must also be undertaken so that the students can actually learn about their total working capacity and get better at managing the product owner's expectations.

6.3.7 Lesson 20: remote demonstrations do not just happen: Unfortunately, the students did not give any of the required product demonstrations during any of the sprint reviews. In all three of the planned demonstrations, the students did not prepare any screenshots or videos for their presentation and the product owner had to compile the code to review the application during these sessions. This impacted the relationship with the product owner and lowered the quality of the feedback that the developers received.

6.3.8 Recommendation 20: if it is important to do something, provide the students with guidelines: Regular interactive feedback sessions with the product owner, coupled with the availability of working software for them to examine, are the cornerstones of agile development processes. They help the client to see the evolving software system directly, leading to more timely and effective feedback. It can also be motivating for everyone working on the project to see signs of progress. Instructors need to emphasise the role of these sessions and provide explicit guidelines for their preparation and content. With the change in the software development process this year, there were many new and somewhat assumed skills that the students were faced with undertaking. Every change in your model will bring new surprises, and instructors need to be more careful in recognising what these will be and in preparing students for these.

7 Conclusions

The Pace University annual global software development education initiative did not stop in 2009; it also took place in 2010 when three extended global teams (43 students in total) persisted with the agile process to develop mobile applications on the theme of *Improving Students' Life on Pace University Campus*, each for a different mobile platform and each with the support of a different end-to-end tooling infrastructure. We reverted back to co-located developers given the many synchronisation issues of 2009, and with no clear and immediate way to overcome them in place. Instead, we examined the use of different tools and the role of quality assurance activities in an agile process context and reintroduced some of the earlier recommendations. We re-introduced the role of auditor and we added the role of external tester to improve the quality of the mobile applications. The extended global teams were composed of developers in the USA, auditors in the USA and testers in Cambodia, India and Senegal. Building upon our annual lessons and recommendations, we intend for future iterations of this education initiative to continue to evolve and, in the process, better serve the emerging needs of our software engineering students.

What is clear is that establishing a working relationship among instructors and creating a student-centred learning environment across educational establishments in different countries are activities that are subject to the same costs as initiating any offshore outsourcing endeavour. The purpose of consolidating the lessons and recommendations from five years of our education initiative in one place (summarised in Table 1) is to show the rationale for the trajectory of our model. Through so doing, we hope to provide background information that can assist other instructors and institutions in their planning and management of global software development education initiatives for students. While we do not offer a prescription for these other instructors, we offer some key points for getting started:

- Start small and only scale up when you have a working model in place. Be fully prepared for the workload because you or your colleagues cannot quit halfway through.
- Focus upon a single project topic initially, not on many disparate project themes, because your attention will need to be on the process rather than on the content of different potential software systems. Putting student development teams in competition on the same project is one way to concentrate on the process and to maximise your early investment, even though this will require some of its own considerations of course.

Table 1 Evolution of the global software development education initiative (2005–2009)

Year	Institutions and roles	Lessons	Recommendations
2005	• Institute of Technology of Cambodia (Cambodia) – clients and end-users • Pace University (USA) – developers	1. It is easy to overlook the costs in start-up, set-up and on-going management 2. Be aware of the lag between theory and practice 3. Maintaining objectivity is difficult, but essential 4. Do not assume that soft skills come easy	1. Attend to project management in student projects as rigorously as industry projects do 2. Account for just-in-time learning and capitalise upon existing expertise 3. Establish independent third-party oversight 4. Create roles and simple protocols to nurture soft skills
2006	• Institute of Technology of Cambodia (Cambodia) – clients and acceptance testers • Pace University (USA) – developers and lead contractors • University of Delhi (India) – third-party suppliers	5. Assumptions about team members materialise from day one 6. If you do it for them, then they do not learn 7. Competition is not always perceived as fair 8. Do not try to run disparate projects with the latest technologies until the underlying process works	5. Focus more than you expect upon social bonding activities and communication protocols, and from day one 6. Establish support systems for both students and instructors 7. If you want healthy competition to improve quality, plan for it 8. Focus upon the process before the project topics and tools
2007	• Institute of Technology of Cambodia (Cambodia) – clients and acceptance testers • Pace University (USA) – developers, lead contractors, mentors and auditors • University of Delhi (India) – third-party suppliers	9. Students experience tunnel vision 10. Friends commit to each other 11. Mentoring or coaching is the next best thing to apprenticeship 12. Effective auditing demands cycle time	9. Push critical project information to students 10. Establish 'official' international community relationship managers 11. Find a way to engage external students as the instructors' partners 12. Factor in time to act upon audits
2008	• Institute of Technology of Cambodia (Cambodia) – clients and developers • Pace University (USA) – developers, quality coaches and quality auditors • University of Delhi (India) – developers • Mahidol University (Thailand) – developers • Royal University of Phnom Penh (Cambodia) – socialisation leaders	13. You can scale up if you have the basic model in place 14. Greater awareness of extended team members matters 15. To win the quality war you may lose some battles 16. It is not sufficient to assign a real-life project; it is also important to make it deployable and sustainable	13. If you change too many things, scale the model back down 14. Run periodic exercises or games so that students get to know each other 15. Monitor the 'health' of all team members regularly 16. Attend to non-functional requirements and secure in-country champions for ease of deployment
2009	• Pace University (USA) – developer and Scrum Master for one sprint • University of Delhi (India) – developers and Scrum Masters for one sprint • Ecole Supérieure Polytechnique (Senegal) – developers and Scrum Masters for one sprint	17. Active training is more effective than passive tutorials and videos 18. Progress meetings need to be done regularly to be effective 19. Tasks need to be of a granularity that promotes estimation and completion 20. Remote demonstrations do not just happen	17. Always facilitate any preparatory work expected of students 18. Use simple calendar, scheduling and notification tools to assist meeting logistics 19. Review planned tasks and help students improve upon their global estimating skills 20. If it is important to do something, provide the students with guidelines

• If you can, run the project after the students have undertaken a preliminary course in software engineering that has taught them about the basic techniques, processes and technologies that they will need to employ. Where this is not possible, ensure that there is a lead-time in the teaching so that the students have the time to plan ahead and practise these skills first. Either way, you are going to have to help them significantly with realistic planning and timelines, templates and guidelines. Prepare a checklist of all the skills that they are going to need and know both when and how these are going to be introduced.

• Distribute the stakeholder roles at first for an easier model (i.e. clients in one country and developers in another country). Distributing the actual development work demands a very careful alignment of schedules for work integration, and maturity in the process and supporting technologies. Consider this arrangement only when you are confident of all these things, but do expect such development work to actually take longer the first time due to the communication overheads.

- Attend to the client/developer relations and provide explicit guidelines on expectations, communication protocols and information sharing. The better this relationship is, the smoother your project will be, so invest time in undertaking regular activities that enable your students to get to know about each other.
- Integrate additional students in supporting roles (e.g. graduate software engineering students for quality assurance and coaching activities, management students for project management activities, and perhaps even marketing students for international relationship management, etc.). They too will need explicit guidelines and support so as to achieve their own learning objectives, but they have the potential to remove a 24/7 demand on you and your colleagues.
- Factor in more cycle time than you think that the students need for receiving and accommodating feedback. There is little point in undertaking reviews, audits, demonstrations and the like if you do not give the students the opportunity to learn and improve. If you cannot accommodate this, then think about reducing the scope in some way.
- Secure a third-party in an oversight role, such as another faculty member or an industry colleague. As you get immersed in the day-to-day matters of your project, you will be focused upon your students, their project work and the accomplishment of their learning objectives. You will value the critical insight that only an external and objective perspective can bring to improve upon your processes.

Finally, it is worth pointing out that while many of the lessons and recommendations in this paper are specific to students and an educational context, some may equally apply to commercial global software development. For example, lesson/recommendation 1 is a stark reminder of the need to do a thorough and through-life cost–benefit analysis before transitioning to this global mode of working. Moreover, it is always going to be prudent to start small with less critical projects (lesson/recommendation 13) and to secure champions (lesson/recommendation 16) when attempting to change working practices in a commercial reality. Similarly important to a successful transition will be the need to establish clear roles and processes across global constituents (lesson/recommendation 8), capitalising upon country-specific expertise where possible. Also, it will be essential to institute any necessary technical and non-technical training programmes at the different locales (lessons/recommendations 2, 4 and 17), as well as to set up support systems (lessons/recommendations 6 and 11), provide suitable guidance materials (lesson/recommendation 20) and establish oversight practices (lessons/recommendations 3 and 12) to aid in the smooth implementation of inter-locking processes. Work breakdown, task allocation and progress reporting will all need particular attention in a newly distributed setting, and can be supported by carefully chosen, albeit simple, shared tools (lessons/recommendations 18 and 19).

Our efforts to improve socialisation within our educational initiative (lessons/recommendations 5 and 14) and to provide dedicated international community relationship managers (lesson/recommendation 10) could likewise prove of value in a commercial setting, although running parallel development efforts to foster quality through competition (lesson/recommendation 7) may not be so viable, except perhaps for those situations in which ultra high reliability of the end products is demanded. Tunnel vision (lesson/recommendation 9) is highly likely to be a factor in any time-sensitive working environment, irrespective of whether educationally or commercially driven, thus enabling the timely dissemination of pertinent information in different culturally acceptable ways is going to be a common challenge to attend to. One would further assume that any effort made to actively monitor the 'health' of all team members working on a global software development project would be invaluable (lesson/recommendation 15), especially if the goal is to facilitate its overall improvement.

8 Acknowledgments

This work was 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), and by an IBM Jazz Innovation Award 'Toward an end-to-end tooling solution with Jazz' (2009). We would like to thank all the students who participated in this education initiative. We are also grateful to the instructors who participated in this project over the years: Thanwadee Sunetnanta, Moniphal Say, Des Phal, Jean-Marie Preira, James Tamgno, Longchrea Neak, and Sopheap Seng.

9 References

1 Naur, P., Randell, B. (Eds.): 'Software Engineering: Report of a Conference Sponsored by the NATO Science Committee, Garmisch, Germany, 7–11 October 1968' (Scientific Affairs Division, NATO, Brussels, 1969). Available at: http://homepages.cs.ncl.ac.uk/brian.randell/NATO, accessed September 2011

2 Royce, W.W.: 'Managing the development of large software systems'. Technical Papers of Western Electronic Show and Convention (WesCon), Los Angeles, USA, 25–28 August 1970, pp. 1–9

3 Boehm, B.: 'A spiral model of software development and enhancement', *IEEE Comput.*, 1988, **21**, (5), pp. 61–72

4 Rational Software: 'Rational unified process: best practices for software development teams'. Rational Software White Paper TP026B, 1998. Available at: http://www.ibm.com/developerworks/rational/library/253.html, accessed September 2011

5 Beck, K., Beedle, M., van Bennekum, A., *et al.*: 'Manifesto for agile software development', The Agile Alliance, 2001. Available at: http://agilemanifesto.org/, accessed September 2011

6 Aspray W., Mayadas F., Vardi M.Y. (Eds.): 'Globalization and offshoring of software: a report of the ACM job migration task force' (Association for Computing Machinery, 2006). Available at: http://www.acm.org/globalizationreport/, accessed September 2011

7 Mockus, A., Herbsleb, J.: 'Challenges of global software development'. Proc. Seventh Int. Symp. on Software Metrics (Metrics 2001), London, UK, 4–6 April 2001, pp. 182–184

8 Damian, D.E., Lanubile, F., Oppenheimer, H.L.: 'Addressing the challenges of software industry globalization: the workshop on global software development'. Proc. 25th Int. Conf. on Software Engineering (ICSE 2003), Portland, Oregon, USA, 3–10 May 2003, pp. 793–794

9 Cramton, C.D., Webber, S.S.: 'Relationships among geographic dispersion, team processes, and effectiveness in software development work teams', *J. Business Res.*, 2005, **58**, (6), pp. 758–765

10 Casey, V.: 'Software testing and global industry: future paradigms' (Cambridge Scholars Publishing, UK, January 2009)

11 Lutz, B.: 'Linguistic challenges in global software development: lessons learned in an international SW development division'. Proc. Fourth IEEE Int. Conf. on Global Software Engineering (ICGSE 2009), Limerick, Ireland, 13–16 July 2009, pp. 249–253

12 Deshpande, S., Richardson, I., Casey, V., Beecham, S.: 'Culture in global software development – a weakness or strength?'. Proc. Fifth IEEE Int. Conf. on Global Software Engineering (ICGSE 2010), Princeton, New Jersey, USA, 23–26 August 2010, pp. 67–76

13 Purvis, M., Purvis, M., Cranefield, S.: 'Educational experiences from a global software engineering (GSE) project'. Proc. Sixth Conf. on Australasian Computing Education (ACE 2004), Australian Computer Society, Inc., vol. 30, pp. 269–275

14 Damian, D., Hadwin, A., Al-Ani, B.: 'Instructional design and assessment strategies for teaching global software development: a framework'. Proc. 28th Int. Conf. on Software Engineering (ICSE 2006), Shanghai, China, 20–28 May 2006, pp. 685–690

15 Petkovic, D., Thompson, G., Todtenhoefer, R.: 'Teaching practical software engineering and global software engineering: evaluation and comparison'. Proc. 11th Annual SIGCSE Conf. on Innovation and Technology in Computer Science Education (ITICSE 2006), Bologna, Italy, 26–28 June 2006, pp. 294–298

16 Richardson, I., Milewski, A.E., Mullick, N., Keil, P.: 'Distributed development: an education perspective on the global studio project'. Proc. 28th Int. Conf. on Software Engineering (ICSE 2006), Shanghai, China, 20–28 May 2006, pp. 679–684

17 Urdangarin, R., Fernandes, P., Avritzer, A., Paulish, D.: 'Experiences with agile practices in the global studio project'. Proc. Third IEEE Int. Conf. on Global Software Engineering (ICGSE 2008), Bangalore, India, 17–20 August 2008, pp. 77–86

18 Monasor, M.J., Vizcaíno, A., Piattini, M., Caballero, I.: 'Preparing students and engineers for global software development: a systematic review'. Proc. Fifth IEEE Int. Conf. on Global Software Engineering (ICGSE 2010), Princeton, New Jersey, USA, 23–26 August 2010, pp. 177–186

19 Gotel, O., Scharff, C., Seng, S.: 'Preparing computer science students for global software development'. Proc. 36th IEEE Annual Frontiers in Education Conf. Borders: International, Social and Cultural (FIE 2006), San Diego, California, USA, 28–31 October 2006

20 Gotel, O., Kulkarni, V., Neak, L., Scharff, C., Seng, S.: 'Introducing global supply chains into software engineering education'. Proc. First Int. Conf. on Software Engineering Approaches for Offshore and Outsourced Development (SEAFOOD 2007), Zurich, Switzerland, 5–6 February 2007, (*LNCS*, **4716**), pp. 44–58

21 Gotel, O., Kulkarni, V., Scharff, C., Neak, L.: 'Students as partners and students as mentors: an educational model for quality assurance in global software development'. Proc. Second Int. Conf. on Software Engineering Approaches for Offshore and Outsourced Development (SEAFOOD 2008), Zurich, Switzerland, 3–4 July 2008, (*LNBIP*), **16**, pp. 90–106

22 Gotel, O., Kulkarni, V., Say, M., Scharff, C., Sunetnanta, T.: 'A global and competition-based model for fostering technical and soft skills in software engineering education'. Proc. 22nd IEEE Conf. on Software Engineering Education and Training (CSEE&T 2009), Hyderabad, India, 17–19 February 2009, pp. 271–278

23 Scharff, C., Gotel, O., Kulkarni, V.: 'Transitioning to distributed development in students' global software development projects: the role of agile methodologies and end-to-end tooling'. Proc. Fifth Int. Conf. on Software Engineering Advances (ICSEA 2010), Nice, France, 22–27 August 2010, pp. 388–394

24 Zweben, S.: '2003–2004 Taulbee Survey. Record PhD production on the horizon; undergraduate enrolments continue in decline', *Comput. Res. News*, 2005, **17**, (3), Available at: http://archive.cra.org/CRN/articles/may05/taulbee.html, accessed September 2011, pp. 7–15

25 Davies, P.: 'What's this India business?: Offshoring, outsourcing and the global services revolution' (Intercultural Press, March 2005)

26 Gotel, O., Kulkarni, V., Phal, D., Say, M., Scharff, C., Sunetnanta, T.: 'Evolving an infrastructure for student global software development projects: lessons for industry'. Proc. ACM Sponsored Second Indian Software Engineering Conf. (ISEC 2009), Pune, India, 23–26 February 2009, pp. 117–126

27 Gotel, O., Kulkarni, V., Say, M., Scharff, C., Sunetnanta, T.: 'Quality indicators on global software development projects: does "getting to know you really matter?"', *J. Softw. Maintenance Evol.: Res. Pract.*, 2010, doi: 10.1002/smr.474

28 Deemer, P., Benefield, G., Larman, C., Vodde, B.: 'The scrum primer: an introduction to agile project management with scrum', Version 1.2. Available at: http://www.goodagile.com/scrumprimer/scrumprimer.pdf, accessed September 2011