Integration Starts on Day One in Global Software Development Projects

Olly Gotel[1], Vidya Kulkarni[2], Christelle Scharff[1], Longchrea Neak[3]
[1]*Pace University, New York, NY, USA, {ogotel; cscharff}@pace.edu*
[2]*University of Delhi, Delhi, India, vkulkarni@cs.du.ac.in*
[3]*Institute of Technology of Cambodia, Phnom Penh, Cambodia, longchrea@itc.edu.kh*

Abstract

Since 2005, Pace University, Delhi University and the Institute of Technology of Cambodia have been partnering to offer students the opportunity to work on globally distributed software development projects. The innovative collaborative model has evolved towards an emphasis on technology mashups for development and communication, mentoring and auditing for assuring quality, and team and software integration for right-sourcing. This paper describes a project where students working in sub-teams were required to integrate their sub-components as a single system for a Cambodian environment. Furthermore, a well-defined design sub-component was subject to a competitive bidding process in an attempt to enhance quality though design diversity. The paper reports on our findings and summarizes the dos and don'ts associated with integration. Both team and software integration needs careful attention from day one on a project, a finding that has repercussions for educational and industrial practice.

1. Introduction and Motivation

Within the last decade, advances in technologies and communications to support software development, coupled with economical incentives, have facilitated the global dispersion of activities and the emergence of global software development as a reality. Software development now entails mashups of technologies and is multi-role, multi-zone, multi-site and multi-cultural, with all the ensuing opportunities and challenges such scenarios present. Global teams distributed throughout different continents need to unite and work as one to integrate components and applications. This not only requires reaching a consensus on the collaborative technologies to be used to support development and communication, but sufficient attention to integration planning at many levels.

Three years ago, faculty at Pace University, Delhi University and the Institute of Technology of Cambodia started collaborating to introduce global software development projects in their courses [4,5,6,7]. The setting of the projects was such that Cambodian students acted as clients and testers, US students acted as developers and lead contractors, and Indian students were sub-contractors for a well-defined component of a larger project (for the Cambodian context). During this period, the innovative collaborative model evolved to explore the use and perception of using different mashups of technologies for development and communication [5], and to include a quality assurance focus through mentoring and auditing [4]. In 2007, the thirty-four students distributed across the three countries worked together on a single software development project to be deployed in Cambodia. The wiki of the project is at: *http://atlantis.seidenberg.pace.edu/wiki/gsd2007*. The project was split up into smaller components that needed to be worked on independently and eventually integrated. The goals for orchestrating one large project, as opposed to many small projects, were to: (a) Get the students thinking about integration at disparate levels. The primary motivation was for students to experience the differences in planning and undertaking integration activities in co-located settings and the integration of globally engineered components. In addition, to experience how to plan for the integration of future requirements in subsequent releases of a software; and (b) Get the students distributed across the three continents working as a single team with a shared goal, as opposed to working on fragmented projects with unaligned goals. The intention was to educate students about the importance of integration at a more fundamental and social level, both locally and globally, as a prerequisite for achieving technical integration.

Giving students such global software development experiences has become a growing trend [2,3]. Less attention has been paid, however, to multi-site, multi-

cultural set-ups that focus on getting the students to work on a single project. Our project is a first attempt at running a single distributed project for students across countries that are positioned at completely different levels on the offshore outsourcing field, a setting that presents very unique integration challenges.

Section 2 provides background to our 2007 project setting. The points of integration we were concerned with exploring are presented in Section 3. Section 4 shows our findings from the experience and Section 5 determines a list of dos and don'ts for integration planning and practice on projects of this nature.

2. Context

This section provides a brief synopsis of the 2007 project set up to provide context for this paper.

Students and courses. The project involved: (a) thirteen fourth year Computer Science students from the *Institute of Technology of Cambodia* (ITC) (*http://www.itc.edu.kh*) taking a Software Engineering course; (b) eight junior and senior Computer Science students from *Pace University* (*http://www.pace.edu*) taking their capstone Software Engineering course; (c) six second year Master of Computer Applications students studying a Database Applications course at the *University of Delhi* (*http://www.du.ac.in*); and (d) seven graduate students from a Masters Program in Software Design and Engineering taking a Software Quality Assurance (SQA) course at Pace University.

Software to be developed. The students were to collaboratively develop a web-based application, called MultiLIB, for the management of the library of the Department of Computer Science at ITC. MultiLIB was to be used by guests, students, professors, the secretary/librarian and administrators. It was to replace an existing Excel-based system.

Sub-systems. The development effort was partitioned in the following way (a decision imposed by the Cambodian instructor): (a) *Librarian/ Administrator side* – for management of the library policies, resources, accounts and loan transactions; (b) *Guest/Student/ Professor side* – to view, reserve, rate and recommend resources and to consult accounts' status; and (c) *Innovation side* – to view electronic resources, such as e-books, audio and video. This latter side was to envision innovative features for version 2.0 of MultiLIB to account for other media.

Teams and Responsibilities. The Cambodian students, split into three sub-teams, four students to work on the student side, five on the librarian side and four on the innovation side, acted as the *clients* and *testers*. They were to explore and validate the requirements, undertake user acceptance testing, and interact closely with the US sub-teams for iteration. The US undergraduates, split into two sub-teams of four students, one to work on the librarian side and the other on the student side, acted as *developers* and *lead contractors.* They worked separately for the two core sides at first and then together during integration. They captured and managed the requirements, proposed design options, managed a Request for Proposal (RFP) that involved subcontracting the database design to the Indian students to leverage their expertise, implemented the software and tested it. The Indian students acted as *third-party suppliers*. The three Indian sub-teams of two students submitted separate bids for the outsourced database component and subsequently collaborated on the selected bid. Moreover, outside the initial intent of the project, they decided to develop their own variant of the software that was then also assessed by the Cambodian client and compared with the US software. The US graduate students acted as *SQA/integration mentors* and *SQA auditors* to help improve and assure the quality of the US undergraduate students' work. They provided coaching sessions on software engineering practices, and reviewed the artifacts delivered and the processes used to deliver them [4]. One US graduate student was assigned to each US sub-team to act as a mentor. A third graduate student was selected as *integration mentor*. The auditors were organized into pairs and assigned to the two US sub-teams.

Process and technologies. All the students followed the same loose waterfall development process with feedback and iteration. The client was continuously available for requirements clarification. While the three institutions did not have aligned semesters, they synchronized on the same milestones with two weeks of setup and team bonding concurrent with six weeks for requirements, four weeks for design, and four weeks for coding and testing. The tooling converged on Eclipse (with JUnit and Subversion) as the development platform, and java.net for bug tracking. The communication tooling comprised six mailing lists (one for each side of MultiLIB and one for each RFP), chats, blogs and wikis (one for each side of the MulitLIB and one for integration). Wikis, editable by US and Cambodian students, contained all the documents and artifacts produced by the sub-teams, e.g., process, requirements, design, testing plans, and code. Wikis were considered as coordination backbone of the project that increased mutual understanding productivity, awareness, and quality on the project [5].

3. One System, Multiple Integration Points

Three types of technical integration activity were planned for this project, in addition to the underlying social integration that would support this.

Integration of the two core sides. The US undergraduate sub-teams began by working with the corresponding Cambodian sub-team to prepare requirements for each core side of MultiLIB. Two requirements documents were produced. They then needed to work together to propose architectural designs and to instrument the sub-contracting process. The intention was for the sub-teams to fuse their work, yet champion the functionality related to their side of the project within an integrated framework.

Integration of a third-party supplied sub-component. The US students distributed RFP letters with timelines, expected responses and selection criteria to the three Indian sub-teams. The response was to include a proposed database design for MultiLIB, including entity relationship diagrams, associated MySQL SQL scripts and sample data. The Indian students were directed to the project wiki for the latest version of the requirements document to prepare their response. The RFP process allowed for interactions with the US project leader for clarification, an opportunity that was underestimated by the Indian teams. The responses to the RFP were evaluated, jointly by the project leaders of the US sub-teams, and either acceptance or rejection letters (with justifications) were returned. The accepted design was refined and resubmitted to the US students by the now integrated team of six Indian students. The US students were to integrate this work into their own.

Integration of future requirements. The Cambodian Innovation sub-team was to explore the latest technologies to disseminate and view electronic assets and so to identify future requirements for MultiLIB. The other two sub-teams were to keep abreast of this work to ensure that the requirements and designs they were proposing would be able to account for anticipated future requirements.

Social integration. Culture is often reported as a challenge in global software development, so supporting the social side of such projects has received much recent attention [1,9]. We therefore invested in socialization to encourage integration at a social level. Country-specific gifts were exchanged between the Cambodian and US undergraduates, the US graduates and the Cambodian students, and the US undergraduates and Indian students. US and Cambodian students exchanged videos about their lives as students. Considerable energy was spent at the beginning of the project to create a bond between the students in these two countries. No such attention was paid to the Indian/Cambodian relationship; this was intended to be invisible to the clients.

4. Findings

Our findings for each of these integration activities are discussed here. We used diverse survey instruments to get feedback from the students and we quote some of their pertinent responses below.

If you don't start together, you don't finish together, irrespective of location. The US students started the project as two local sub-teams and worked independently on the requirements. However, the stakeholders, library policy, items of the library and non-functional requirements were common to the two sub-teams. Starting the design phase, the students realized that the overall architecture of the system had to be common, the user interface had to be uniform, and the RFP was to be managed and decided upon jointly. At this point, to facilitate integration, a mailing list (including all the US and Cambodian students) was created, and a shared integration wiki for common artifacts was created. There was little trust between members of the original sub-teams in the quality of each other's work, a problem compounded by the inability to meet as a whole team outside of class time: "*Even at the end of the project specific members refused to collaborate with what the team, as a whole, decided in the first place"*. Integration at a local level is an uphill struggle if the members do not start off as a united team from day one.

Future-proofing does not happen in the future, it has to we worked in from the start. More globally, integration to account for the work of the Cambodian innovation sub-team did not happen at all. While the US students attempted to keep abreast of what this sub-team was doing initially, this focus began to slip as managing the above integration aspects became paramount. Even though future requirements could be anticipated from the Cambodian work, they were eluded for the sake of expediency. This activity should have been given a more predominant responsibility.

If you want to integrate it, document it. The US sub-team leaders were the main actors in the RFP evaluation process. They admitted that they were not the database experts in their sub-teams, but they were the most motivated members on the sub-teams and wanted to be able to respect the deadline for sending acceptance/rejection letters. The US students verified that all the deliverables were provided, compared the three data models, and weighed the pros and cons of

each to inform their final selection. They accepted the design that was: "*well-organized, contained the required MySQL code, and was well-documented*". Considering their experience, the Indian students said they: "*felt like real sub-contractors who wanted to get a contract from a foreign company*".

Design diversity can lead to better designs, but lack of trust in the process fracture constituents. Because of time and implementation constraints, the US students went on to design and develop a simplified database design based on the three original Indian designs. They learned the value of exploring design options and different perspectives prior to converging on a solution. However, it led to the following passionate responses from the Indian students: "*I would like to know why they did not integrate our database design as they themselves found it the best*"; "*This is strange and unfair*"; "*This is quite unprofessional and moreover if they had any issues then we should have been made aware of it so that we could have given our support"; "It is difficult to understand the design of other students. But it is a kind of violation of the whole concept of global software development"*. It can be difficult to teach students to delegate; to relinquish some control requires not only trust, but also a prepared framework in which to integrate the contribution. The consequence was that these negative feelings triggered a competitive situation and a stronger sense of unity in the Indian team.

The team that communicates, respects each other and shares a purpose, integrates. The six Indian students organized themselves to develop a separate version of MultiLIB, such that two students worked on the guest/student side, two worked on the professor side and two worked on the librarian/administrator side. They decomposed the work this way because they felt an imbalance in the two original core sides. The Indian students preferred working with two separate requirements documents: "*It was good to have two separate requirements documents since it helped [identify] the requirements more clearly. But at times we faced some issues regarding the inconsistency in the common parts of the two documentss*". The differences between the way in which the US and Indian students were introduced to the project impacted the teams' unity.

Integration potential deteriorates with communication bottlenecks. The fact that the students were all working on one single project, were dependent upon each other and operating according to different schedules, meant that answers to questions and feedback on work was not always as timely as expected. For instance, the Indian students were waiting anxiously for the requirements document. They experienced delays in getting answers to questions sent by email when mediated through the US students and, since some doubts they had were not resolved entirely, they had to make assumptions about the database design. This situation reflects the difficulty developers experience when working solely with written requirements in isolation from the client.

Integration requires a backbone. The sub-teams were provided with separate wikis at the start of the project for instructor visibility. These wikis served as the communication backbone on the global project [5]. However, attempting to create a specific shared integration wiki later in the project to share project materials was a good step too late. Although updated by the US students on the shared wiki, the Indian students waited for emails regarding any changes and did not check the shared wiki regularly. Shared resources do not imply shared awareness. The technology support for projects of this nature need careful design and process education before the first document on the project is even drafted.

Integration requires a champion. The local integration of the student and librarian sides of MultiLib proved more challenging in the US than in India. This was partially due to competition between the two US sub-teams arising from the initial sub-division. For instance, they competed on the user interface look and feel. Also, they would occasionally neglect to notify each other of changes. The two project leaders, helped by the mentors, did an admirable job of the overall system integration, but this should have been explicitly tasked and planned from day one. Integration needs its leader. The Indian students, in contrast, worked together in a harmonious way with shared responsibilities.

Team integration must be nurtured and sustained at the social level; trust comes from respect. Students on this project had difficulties in trusting each other at times. The distributed sets of students considered themselves more as local teams than as a single global team. Moreover, the co-located US teams found it difficult to re-establish themselves as a single local team after having started out with separate purposes, whereas the Indian team united even after having started out as competition rivals. The Cambodian client teams never fully engaged with the US development team after the requirements phase had completed. With two sub-teams competing for their approval and seemingly not always providing a united from to the client, issues surrounding perceived respect came to the forefront. The lack of communication during the RFP process prevented the Indian team to

integrate with the US team. Despite some of the challenges, the students all cited a positive experience: "*Learning about Cambodia was the best thing for me. I love learning about different cultures and I really think I will stay friends with the students from Cambodia for a long time*". We created the right environment, but we did not sustain the effort; what the individual students took away was directly proportional to what they individually put in.

5. Integration Dos and Don'ts

In this section, we consolidate our findings into a table of dos and don'ts for integration in global software development projects (see Table 1).

Table 1. Integration Lessons.

DO	DON'T
Plan integration from day one (or preferably much earlier).	Underestimate integration issues and think it can be introduced on the fly.
Focus on the team as a whole and on the overall architecture of the system before dividing the work.	Divide the work without first creating the larger team and environment in which these contributions will play a role.
Invest on socialization for team cohesion, e.g., scheduled chats, exchange of gifs, and announcements of respective holidays.	Tolerate disrespect based on ignorance and differences.
Ensure awareness through efficient communication, e.g., mailing lists and wikis.	Use communication silos, e.g. different wikis that eventually need to be integrated.
Publicize the role of everybody on the project to create an environment of equal partnership.	Isolate anybody and ignore the feelings of others.
Use of a common set of consensual technologies across location.	Impose technologies without account of their perception and without training.
Use a process that involves the client, allocates time for feedback, feedback response and diversity at diverse levels.	Use a process that is very rigid.
Create a trustful environment supporting work delegation.	Think one person can do it all.
Have integration leaders for development, communication and socialization.	Misjudge the importance of integration leadership at disparate levels.
Clearly and concisely document the sub-components, interfaces and the integration process.	Go blind without documentation of the artifacts and integration process.

6. Conclusions

Integration, whatever the context and level, is not easy. By its very definition, it implies the bringing together of disparate pieces into a whole. The factors that serve to either support or jeopardize this process in the context of global software development need to be articulated and shared. Most of these factors are activities that need to be attended to from day one (preferably earlier). This paper has been an attempt to provide an initial list of integration dos and don't for educational and industrial settings based on the experiences of running an integration-critical project.

7. Acknowledgements

This work is supported by a National Collegiate Inventors and Innovators Alliance grant (#3465-06), "Incubating the Next Generation of Global Software Development Entrepreneurs" (2006-2008). We thank all the students who were involved in this project.

8. References

[1] Moe, N. B. and Smite D. "Understanding lacking trust in global software teams: A multi-case study". Proc. Conference on Product Focused Software Process Improvement (PROFES 2007), LNCS 4589:20-34 Riga, Latvia, 2007.

[2] Damian, D., Hadwin, A., and Al-Ani, B. "Instructional design and assessment strategies for teaching global software development: a framework". *Proc. Conf.on Software Engineering (ICSE2006)*, Shangai, China, 2006.

[3] Damian, D., B. Al-Ani, D. Cubranic, and L. Robles. "Teaching requirements engineering in global software development: a report on a three-university collaboration, *Proc. Workshop on Requirements Engineering Education and Training*, Paris, France, 2005.

[4] Gotel, O., Kulkarni, V., Neak, L. and Scharff, C.. "Students as Partners and Students as Mentors: An Educational Model for Quality Assurance in Global Software Development". Submitted to *Conf. on Software Engineering Approaches For Offshore and Outsourced Development (SEAFOOD 2008)*.

[5] Gotel, O., Kulkarni, V., Neak, L. and Scharff, C. "Working across borders: Overcoming culturally-based technology challenges in student global software development". *Proc. Conf. on Software Engineering Education and Training (CSEET 2008)*, Charleston, USA, 2008.

[6] Gotel, O., Kulkarni, V., Neak, L., Scharff, C. and Seng, S. "Introducing global supply chains into software engineering education". *Proc. Conf. on Software Engineering Approaches For Offshore and Outsourced Development (SEAFOOD 2007)*, Zurich, Switzerland, 2007.

[7] Gotel, O., Scharff, C. and Seng, S. "Preparing computer science students for global software development". *Proc. IEEE Conf. on Frontiers in Education (FIE 2006),* San Diego, USA, 2006.

[8] Herbsleb, J. D. "Global software engineering: The future of socio-technical coordination". In *Proc. Conf. on Software Engineering – The Future of Software Engineering (ICSE-FASE 2007)*, Minneapolis, USA, 2007.

[9] Proc. of the *Workshop on Supporting the Social Side of Large Scale Software Development (SSSLSSD 2006) at Computer Supported Cooperative Work (CSCW 2006),* Banff, Canada.