Evolving an Infrastructure for Student Global Software Development Projects: Lessons for Industry

Olly Gotel
Pace University
New York, USA
ogotel@pace.edu

Vidya Kulkarni
University of Delhi
Delhi, India
vkulkarni@cs.du.ac.in

Des Phal
Royal University of Phom Penh
Phnom Penh, Cambodia
phaldes@gmail.com

Moniphal Say
Institute of Technology of Cambodia
Phnom Penh, Cambodia
say.moniphal@gmail.com

Christelle Scharff
Pace University
New York, USA
cscharff@pace.edu

Thanwadee Sunetnanta
Mahidol University
Bangkok, Thailand
cctth@mahidol.ac.th

ABSTRACT

With the rapid increase in offshore outsourcing of software development, Global Software Development (GSD) has become the need of the hour. Today's information technology, in the form of communication networks and tooling opportunities, provides us with a supposedly ready infrastructure to support GSD. However, selecting an appropriate combination of tools that cross cultural boundaries and account for unique in-country connectivity situations is not a trivial task. In this paper, we describe our experience of evolving an infrastructure for student GSD projects over a period of four years, culminating in an environment to accommodate the needs of five different teams from four globally dispersed universities in countries straddling many technological divides. We suggest that our experience offers lessons that can also support those organizations embarking upon GSD initiatives and with their own infrastructure decisions to make.

Keywords

Global software development, Infrastructure, Offshore development, Software development tooling, Software engineering education.

1. INTRODUCTION

In 2005, faculty from Pace University in New York City and the Institute of Technology of Cambodia (ITC) initiated a collaboration to enable undergraduate computer science students from the two academic institutions to work together on the development of software systems for the Cambodian market [1]. The Cambodian students acted as clients and the US students acted as developers. The motivation was to have the students gain a real-life experience of offshore outsourcing, and so to begin to understand its potential opportunities and challenges. The model of collaboration was extended in 2006 to include the participation of graduate students from a database course at the University of Delhi in India [2]. The focus was now on incorporating global supply chains and right sourcing into the model, thus exposing the students to the management of sub-contracts and dealing with the communication and coordination issues that may arise from this type of setting. The Indian students therefore acted as sub-contractors for the database design component of each student project. In 2007, the scenario was further refined to emphasize scale and deployment, thereby exposing students to integration planning and resource sharing [3, 5]. The model was such that the US developers worked on separate components of a single software system to be eventually integrated for the Cambodian clients. Teams of Indian sub-contractors responded to a competitive Request For Proposal for the database design, one design of which was to be selected and integrated by the US developers. US graduate students also participated as mentors and auditors in an attempt to highlight quality matters [4]. Lessons learned from these three previous years of collaboration influenced the setting of 2008.

In 2008, teams of developers in Cambodia, India, Thailand and the US were put in friendly competition to re-develop the software of 2007. This was because the Cambodian client did not accept the final software product of 2007 due to unfulfilled requirements. The objective of 2008 was therefore to capitalize upon the work to date and increase the chances of one system being successfully selected for deployment. To address the prior quality issues, partially arising from the fact that trainee software engineering students can be overwhelmed by a myriad of demanding and new tasks on a project of this nature, each global team was assisted by US graduate students acting as software quality assurance (SQA) coaches and auditors. These graduates were tasked to control the quality of the process and final product. In addition, the requirements were re-engineered by the Cambodian clients with the benefit of hindsight and support by dedicated client-side coaches who were experienced in requirements engineering topics. An additional dimension was the incorporation of socialization activities to permit the students to bond during the project, improving engagement to help better achieve a common goal.

This paper outlines the infrastructure that evolved over the four years to bring students with different roles together to work on a Global Software Development (GSD) project of this nature, accounting for different backgrounds and prior exposures to technologies. Section 2 provides the context for the 2008 setting of the project. Section 3 justifies the choice of tools and describes those that were used to support the different engineering, communication, project management and socialization activities in 2008. It also focuses on the technology used to retain oversight of the overall global initiative from the instructors' perspective. Section 4 presents the findings on practical use and impact, and Section 5 presents a consequent list of do's and don'ts when configuring an infrastructure for student GSD projects. Based upon our experiences, we propose a set of simple considerations to guide initial tooling decisions when embarking upon such projects in industrial settings in Section 6.

2. CONTEXT

The students who partnered on this project in 2008 were undergraduate and graduate students in computer science and software design and development from: the Institute of Technology of Cambodia (ITC), the University of Delhi in India, Mahidol University in Thailand, Pace University in the US (New York City NYC and Pleasantville PLV campuses) and the Royal University of Phnom Penh (RUPP) in Cambodia. Table 1 illustrates the roles and responsibilities of the students. Each global development team (five teams overall) was composed of a Cambodian client and a client coach, the development team and a developer coach, and a team of auditors. Each global team was requested to build a version of MultiLIB, a library system for the computer science department of ITC. A socialization team, designed to investigate the impact of educating developers about the client's country and culture on the quality of their interactions with the client, was composed of the US students of Pleasantville (US PLV) and the RUPP students. The total project arrangement is illustrated in Figure 1.

The requirements for MultiLIB were jointly developed by the Cambodian clients and their client-side coaches, and comprised 71 functional and 4 non-functional requirements come the end of the project. It is important to note that the requirements underwent a number of revisions and were permitted to change as development was underway. This meant that each development team needed a way to channel questions to the clients that led to the improvement of the requirements and to implement effective change management procedures to account for requirements changes.

Each development team followed a loose waterfall-based software engineering process with iteration and feedback cycles to account for the unfamiliarity of the developers with software engineering practices, to coordinate the work of students distributed in five different locations and to facilitate the SQA activities of the coaches and auditors. The project spanned nineteen weeks – including four weeks for requirements, four weeks for design, six weeks for coding / testing and five weeks for deployment. The software was delivered and tested incrementally by one-week iterations; developers held planning meetings at the beginning of each iteration where they committed to implement a certain number of requirements.

The client and client coaches selected the software of highest quality, measured in terms of requirements satisfaction. This selection process took place after the completion of the development period and fourteen weeks after project initiation. The selected software was deployed into operations at ITC. The objectives of the initiative were hence realized.

Table 1. Global teams, roles and responsibilities in 2008
(UG and G denote undergraduate and graduate students)

Countries, roles and numbers	**Responsibilities**
Cambodian client (ITC) (5 UG)	Own and manage requirements; Solicit software systems from five development teams; Review and give feedback on all aspects of developers' work; Interact with client coaches and development teams; Test candidate software and select the one of highest quality; Deploy the selected software in Cambodia.
US client coaches (5 G)	*One dedicated coach for each client.* Mentor the client team; Help with the production and maintenance of the requirements document; Manage requirements changes, versioning, change awareness and protocols associated with change management.
Cambodian developers (ITC) (4 UG)	*Five development teams in friendly competition for the development of the highest quality software.* Assess the risks associated with the delivery of a system to satisfy the requirements; Describe the software engineering process to be followed; Propose design options; Iteratively implement, integrate and test software; Reflect on process and communication protocols; Interact with client, mentors and auditors, and integrate their feedback.
Indian developers (5 G)	
Thai developers (5 UG)	
US NYC developers (5 UG)	
US PLV developers (4 UG)	
US developer coaches (5 G)	*One coach for each development team.* Mentor the developers; Provide coaching with techniques and practices introduced in classroom; Raise early concerns to developers and instructors; Formulate a test plan and test the system of the sponsored team.
US auditors (16 G) Thai SQA trainees (4 UG)	*Three US auditors for each development team; One US SQA manager to oversee and coordinate the audit efforts.* Propose an audit plan; Audit and report on deliverables and process to assure and improve quality; Test the system; Interact with developer coaches and developers to raise issues and provide timely feedback. Four Thai SQA trainees observed the quality assurance tasks of the US auditors. The objective of such observation was to establish continuous improvement on testing processes locally.
Cambodian socialization team (RUPP) (2 UG)	Introduce the US PLV development team to many aspects of Cambodian life and culture.

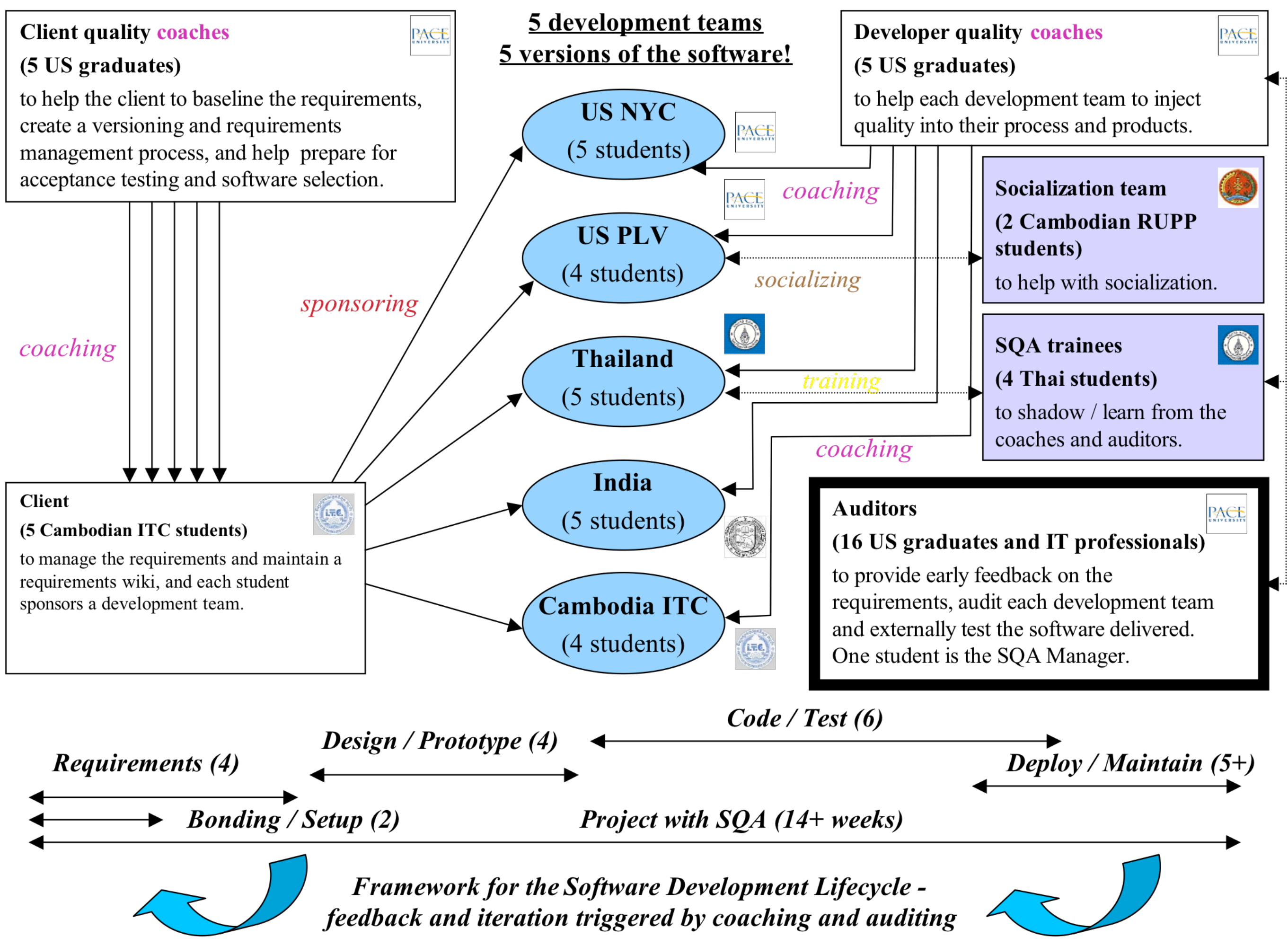
Client quality coaches
(5 US graduates)
to help the client to baseline the requirements, create a versioning and requirements management process, and help prepare for acceptance testing and software selection.
5 development teams
5 versions of the software!
Developer quality coaches
(5 US graduates)
to help each development team to inject quality into their process and products.
US NYC
(5 students)
coaching
US PLV
(4 students)
socializing
Socialization team
(2 Cambodian RUPP students)
to help with socialization.
sponsoring
coaching
Thailand
(5 students)
training
SQA trainees
(4 Thai students)
to shadow / learn from the coaches and auditors.
coaching
India
(5 students)
Client
(5 Cambodian ITC students)
to manage the requirements and maintain a requirements wiki, and each student sponsors a development team.
Cambodia ITC
(4 students)
Auditors
(16 US graduates and IT professionals)
to provide early feedback on the requirements, audit each development team and externally test the software delivered. One student is the SQA Manager.
Code / Test (6)
Design / Prototype (4)
Requirements (4)
Deploy / Maintain (5+)
Bonding / Setup (2)
Project with SQA (14+ weeks)
Framework for the Software Development Lifecycle -
feedback and iteration triggered by coaching and auditing

Figure 1. Overall set-up of the global project in 2008 (note deployment took 5 weeks and maintenance is now on-going)

3. TOOLING ENVIRONMENT

The use of a standardized tooling environment has been emphasized in our collaboration to improve the coordination of the work of the students across locations. The introduction of the tooling has been supported by the design of tutorials and dedicated sessions with hands-on laboratories. The tooling used by students on the project can be categorized into engineering, communication, project management and socialization tooling. Additional tooling was required to support the instructors in their planning and oversight. This section describes the tooling environment and its evolution over four years, and provides justifications for our choices.

3.1 Rationale

Planning a tooling environment that accounts for unique in-country situations and multiple parties is not a trivial undertaking. Based on some initial choices, our mix of tooling has evolved organically over the years depending upon what the students have chosen to use or ignore.

The initiative intentionally selected open source and free tooling to facilitate access by the students across the locations. The tools were also chosen for their widespread use in industry, their relative simplicity of use and the functionality they provide to support necessary distributed GSD practices. They were installed and configured by the instructors. In 2008, the lessons from the three previous years suggested the need to give higher importance to communication, project management and socialization tools. Past experience also indicated the importance of redundancy in communication channels and so the need for different types of communication tools, including emails, chats and wikis. Difficulties were faced in selecting an appropriate UML design tool due to the complexity of the existing tools in this domain. The only design tool that was recommended to the students was DB designer 4 to propose Entity Relationship Diagram (ERD) design options. Java.net was chosen for its ready-to-use subversion and issue tracking facilities.

The evolution of the base tool set over the years is shown in Tables 2a and 2b, and specific details of the tooling for 2008 are elaborated upon in Table 3.

Table 2a. Evolution of tooling (2005-2006)

Tool type	2005	2006
Technical	Eclipse IDE with JUnit, MySQL	Eclipse IDE with JUnit and CVS, MySQL
Commun-ication	Yahoo IM, Yahoo groups, Websites, blogs	Yahoo IM, Yahoo groups, Websites, blogs
Socialization	Yahoo IM, exchange of gifts	Yahoo IM, exchange of gifts
Project management	Website pages, blogs	Website pages, blogs

Table 2b. Evolution of tooling (2007-2008)

Tool type	2007	2008
Technical	Eclipse IDE with JUnit and subversion, MySQL, java.net for issue tracking	Eclipse IDE with Junit and subversion (Netbeans - India; Visual Studio – Thailand), DB designer for database design, MySQL, java.net for issue tracking
Commun-ication	Yahoo IM, Yahoo groups, wikis, blogs, videos	Yahoo IM, Yahoo groups, wikis, wink for video screen captures
Socialization	Videos, Yahoo IM, exchange of gifts, videos	Yahoo IM, Second Life, Socialization exercises
Project management	Wikis, blogs	Wikis, Google calendars, Time-zone software

3.2 Engineering Tooling

The use of an Integrated Development Environment (IDE) has been encouraged from the beginning of this initiative in 2005. Therefore, the engineering tooling environment converged to the use of Eclipse / Netbeans / Visual Studio with JUnit / NUnit for unit testing and subversion for code version control in 2008, given these IDE were integrated in the curriculum at the various institutions and the initiative had matured enough to support flexibility in local development environment. The maturity in relationship and stability of some of the other tooling enabled this to happen. Students therefore built on their programming skills; the Thai students used Visual Studio as the programming language they had mastered to date and they would use on the project was to be C# and ASP .NET, whereas all the other students developed in Java and JSP. The use of subversion was emphasized this year to facilitate integration and incremental code delivery to the coaches and auditors for review and to the client for deployment. Java.net issue tracker was introduced in 2007 and used this year by the developers, coaches and auditors to manage issues and verify the software respectively.

3.3 Communication Tooling

Communication between the students was carried out using emails and chats. Each student created a Yahoo email and pseudo to communicate during the project. In 2008, nine mailing lists were created – one for each development team, one for the client and client coaches, one for the auditors and coaches, one for the socialization team and one for the instructors -- and use was monitored by the instructors to detect early impediments on the project. The local development teams often used phone to communicate, and met face-to-face, but the global teams did not communicate via voice or video even though they were encouraged to do so; the belief was that videos and voice conversations would be too difficult to set up.

3.4 Project Management Tooling

Shared Google calendars and time-zone software were used for sharing milestones and time awareness across locations. The use of wikis as the coordination backbone of the project was introduced in 2007. Wikis were created in place of web pages to provide a standardized environment to permit the contributions of students at each location, to share documents (written and videos) and software artifacts, to increase overall team awareness, assist with SQA activities and facilitate deployment. Eight wikis were created using the same template - one for each development team, one for the client and client coaches, one for the auditors and coaches, and one for the socialization team. These were effectively separate wiki spaces with differing editing permissions within an overarching GSD wiki and all pages were visible to all parties. Blogs that were used by the students from 2005 to 2007 for weekly reflections were abandoned in 2008 and replaced by posting progress and reflections directly on the wiki.

3.5 Socialization Tooling

Photos of all the students were posted on the wikis so that students could put a face to a name. During the first two weeks of the project, students were required to meet in chat sessions to get to know each other and establish the relationships that nurture distributed global teams to work efficiently and gain trust. Participation in these chats was not required of the coaches and auditors, a decision that was detrimental to the team bonding in some of the global teams. After the first few weeks of scheduled chats, students were then free to socialize and have informal chats to sustain the relationships. In an attempt to study the impact of socialization in GSD projects, in a controlled manner, the US PLV team was the only team intensively exposed to the Cambodian culture of the client. In previous years, all the students exchanged videos about themselves and gifts, so this aspect could not be studied in detail. Note that this study is not elaborated on in this paper. These students received country- and city- specific gifts and were required to meet every week with the RUPP students in Second Life to discuss predefined topics related to Cambodia. At the end of the project, all students were invited to the Pace Second Life Island for a party with virtual food, drinks and country-specific music and activities to celebrate their achievements. The party was well attended by the US developers, the Thai and RUPP students, and all the instructors.

Table 3. Project activities and their tooling in 2008

Activity	Tools	Rationale and tasks supported
RFP process	MS Word	To write the RFP for MultiLIB and selection letters with justifications.
	Email	To solicit bids, manage RFP process and select the highest quality system.
	Chat	To manage the RFP process.
	Wiki	To manage the delivery of the artifacts that satisfy the RFP.
Requirements	MS Word	To gather, validate and manage requirements.
	Chats, email, face-to-face	Synchronous and asynchronous communication about requirements.
	Wiki	To publish the requirements document versions and clarify requirements in a FAQ.
Design	HTML, UML and DB designer 4	To facilitate communication on the GUI with the client by proposing a HTML mockup; To model design options (including an Entity Relationship Diagram) and achieve a better understanding of how the system should behave and correspond to client needs.
Implementation	Eclipse / MS Visual studio / Netbeans	To take advantage of the IDE features (e.g., unit testing, versioning, externalization to support different languages, etc.).
	Java / JSP / C# - Tomcat / IIS	To encourage students to build upon their programming skills.
	MySQL	To use an open source DBMS that would be easily deployable in Cambodia.
	Wiki	To manage the iterations and delivery using an iteration table that describes the status of development and testing of each requirement.
Testing	JUnit / Nunit	To automatically run unit tests and validate units of the software.
	Java.net issue tracker	To validate software by each of the developers, developer coaches and auditors, and to report, fix and manage issues.
Configuration management / code delivery	Subversion	To facilitate code sharing, change and version management; To permit code delivery to the client for deployment.
Project management	Wiki	To contain all produced documents and software artifacts; To increase milestone visibility and project awareness; To post weekly reflections to allow problems to be addressed by the teams and instructors.
Coaching	Email	To provide mentoring and feedback on technical and team management aspects.
Auditing	Wiki	To retrieve the artifacts to be audited and check compliance; To post audit results.
	Email	To communicate audit results.
	Face-to-face	To coordinate auditing activities inside the audit teams.
Socialization	Wiki and face ex.	To get to know each other and put a face on a name.
	Map ex.	To get to know each other's country and city.
	Chat	To get to know each other and facilitate spontaneous conversation.
	Second Life	To create a fun environment for socialization activities for US PLV and RUPP students; To celebrate the end of the project by organizing a Second Life party.
Deployment / maintenance	Email	To communicate with developers of the software selected for Cambodian deployment.
	Wiki	To retrieve the artifacts of the selected software and to facilitate maintenance.
	Eclipse, Tomcat, MySQL, LDAP	To deploy the selected software, implement any missing features, provide authentication and improve the delivered system.

3.6 Oversight Tooling

An important part of the project was providing the ability for the seven instructors who had a role to play on this initiative the ability to communicate, plan and track the work. An instructor-specific mailing list and visibility of the overall GSD wiki facilitated these tasks. All the instructors have met in person at one stage or the other. This prior contact smoothed out all communications considerably. The wiki of the project can be found at: http://atlantis.seidenberg.pace.edu/wiki/gsd2008.

4. FINDINGS

Five versions of MultiLIB were implemented. The Cambodian, US PLV, US NYC, Indian and Thai teams implemented 24, 33, 64, 66 and 68 of the functional requirements respectively. The Cambodian client accepted the software of India, Thailand and NYC and was hard pushed to select an overall winner. The final selection was based upon the additional satisfaction of non-functional requirements and consideration of the quality of the team's software development process (as determined by the auditors). Effective use of the tooling was largely responsible for the achievements of the US NYC, Indian and Thai teams.

US PLV and Cambodia both had unique challenges not faced by the other teams. The PLV developers were required to learn additional technologies as per demands for socialization. The Cambodian developers faced a number of power cuts and consequent server problems. Both teams experienced delays to their progress due to technology choices and environments that they only began to master too late in the project.

4.1 Tool Use

Students were new to many of the technologies introduced. Based on the exit survey, the Indian and Thai developers perceived the IDE as the most difficult technology to learn; the US developers had difficulties with the design tools and the Cambodian students found java.net too complex. The developers unanimously designated the IDE as the 'killer' engineering tool though. Yahoo IM was designated the 'killer' communication tool for Cambodian and US NYC developers whereas emails were crucial for the other developers. All developers were asked to name the two most crucial tools on the project overall. All the teams cited an engineering and communication tool combination, except for the Indian developers who named their IDE and java.net – pure technology. A summary of the students' exit survey results is given in Table 4.

Subversion and unit testing were used only in a limited manner. Students found it difficult to become familiar with continuous code integration and regression testing practices while coping with the demands of a global project context. The number of revisions in subversion was low, ranging from 13 (Thailand) to 29 (US-NYC) revisions, and the number of developers who committed code ranged from 1 (India, Thailand) to 2 (US) – low numbers that contradict with the philosophy of use for such tools. There is a need for more context and process-related training with such tools, and shareable tutoring technologies could help somewhat here. Java.net issue tracker was used effectively to submit bugs by the auditors and developer coaches, and to manage bugs by the developers in India, Thailand and US NYC.

Communication problems between the client and the developers arose, not only from the difficulties of access to the Internet of the Cambodian students, but also from the differences in times, semester and vacation misalignment, and class loads. A comparative summary of the use of the two major channels of client / developer communication (email / mailing lists for asynchronous communication and chat for synchronous communication) is shown in Figures 2 and 3. These data were extracted from records of mailings and postings of chat records to the development team wikis.

All teams were heavy users of the mailing lists. An email to their mailing list distributed information to the entire extended global team, including the client project sponsor. However, the use of synchronous communication was minor by comparison; students found it difficult to set up convenient times across time-zones to chat. What is interesting is that the three teams that made the most additional use of synchronous channels to communicate with the client are notably those who completed the software to the client's satisfaction. Figure 4 illustrates how it is not the quantity of communication that matters, but the blend of modes. Synchronicity can really help address points of misunderstanding and keep a project moving.

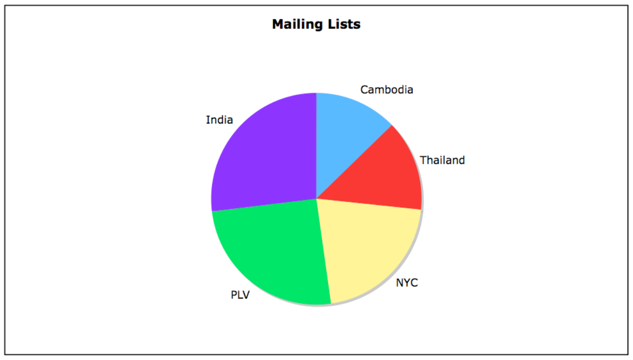

Figure 2. Asynchronous communication between clients and development teams (comparative)

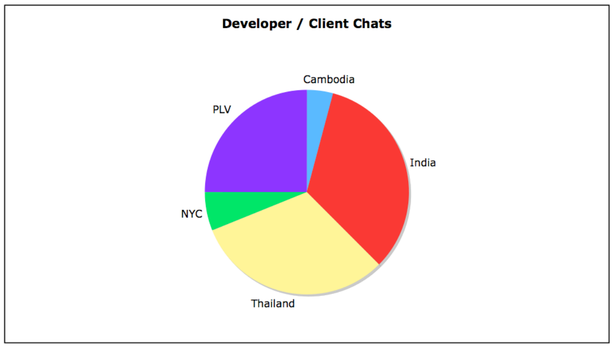

Figure 3. Synchronous communication between clients and development teams (comparative)

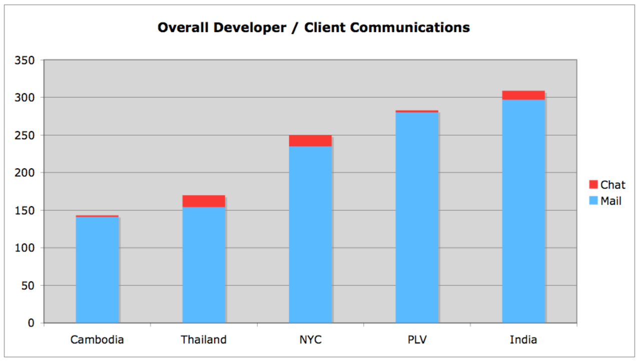

Figure 4. Blend of synchronous and asynchronous communication between clients and development teams (comparative)

Communications between the US-based developers and auditors was not as effective as it could have been possibly because the team bonding was not created at the beginning of the semester (overlooked since co-located) and because problems with the localization of deliverables on the wiki frustrated some of the auditors. Milestone deliveries were typically not announced on the mailing lists by some of the development teams because the developers assumed shared awareness of calendars. One setup of the wiki that would have been helpful was the notification of any new post on the wiki, but there is obviously also a trade-off so as not to overwhelm. Additionally, auditors had to maintain the auditors' wiki and the wiki of the developers they audited with some redundancy. The coaches had more communications with the developers and clients due to their mentoring roles.

In the past years, we found that socialization provides a foundation for our GSD projects, but attention to such efforts declines amongst the students with the pressure of approaching deadlines. The relationship created by establishing a regular meeting of the socialization team in Second Life permitted the US PLV students to feel more comfortable with the client and establish a friendly relationship. However, this did not have the positive impact on local team bonding and quality that was anticipated. One explanation was simply that the demand to learn new technology on this project is already considerable for students and one extra seemingly tangential technology can be one too many unless carefully accommodated in the learning process. An interesting finding for socialization though was the enthusiasm of the participants in the Second Life party at the end of the semester, which positions Second Life as a promising tool for socialization on GSD projects if used at the right junctures.

Table 5 summarizes the extent of the use of the various tools, categorized as High, Medium or Low (H, M, L), again in a comparative rather than absolute manner. Empty cells indicate that the tool was not used because it was either not required for the role or due to the difficulties in getting up to speed with the technology in courses or locations. Tools used during the audit reviews (R), deployment (D) and the Second Life party (P), are classified by R, D and P respectively in Table 5.

Table 4. Summarized results of tool use in 2008 (from the student exit surveys)

Team	**What were the new technologies?**	**Most difficult technology to learn?**	**Killer technical tool?**	**Killer communication tool?**	**Two most crucial tools (overall)?**
Developers - Cambodia	IDE, design tools, subversion, Java.net issue tracker, wiki	Java.net	IDE	Yahoo IM	IDE / Emails
Developers - India	IDE, design tools, subversion, Java.net issue tracker, wiki	IDE	IDE	Emails	IDE / Java.net issue tracker
Developers – Thailand	IDE, design tools, DBMS, subversion, Java.net issue tracker, wiki	IDE	IDE	Emails	IDE / Emails
Developers – US NYC	Design tools, DBMS, subversion, Java.net issue tracker, wiki	Design tools	IDE	Yahoo IM	Emails / IDE
Developer – US PLV	Design tools, DBMS, subversion, Java.net issue tracker, wiki	Design tools	IDE	Emails	IDE / Emails

Table 5. Tooling, roles and actual use on the project

Tool	Client	Client coach	Cambodia	India	Thailand	US NYC	US PLV	Developer coach	Auditors	Socialization team
MS Word	H	H R	H	H	H	H	H	H	H R	
UML / DB Design			H	H	H	H	M	R	R	
IDE	D		M	H	H	H	H	R	R	
MySQL	D		H	H	H	H	H	R	R	
Java / JSP / C#	D		M	H	H	H	H	R	R	
Sub-version	D			M	M	M	M	R	R	
JUnit / Nunit				L	L	L		R	R	
Java.net				H	H	H	L	M	M	
Email	M	H	M	H	M	H	H	M	M	L
Chat	M	H	M	M	H	H	M	M	L	M
Wiki	H	H	M	H	H	H	H	H	H	L
Demo Video			H	H	H	H	H			
Second Life				L (P)	H (P)	M	H			H

4.2 Tool Perceptions

Quotes from the students' exit survey are provided below to illustrate some of their main concerns associated with the tooling and their respective roles on the project. The quotes tend to be in accordance with the findings above.

- Several Cambodian client students stated that they would have preferred: "more activities and less email replies".
- A Cambodian developer reported that: "What I liked most on the GSD project is that it improved my experience such as: knowing the development process, database design, class design, how to implement a database in a DBMS, OOP concepts, new technologies (JSP, Servlets, Java beans, Struts, MVC but only the basics) and working in group."
- The Indian developers mentioned that what they liked the most on the GSD project were: "the new tools that [they] came to know". They did not have issues in learning the new tools and said that it: "hardly took a day to get the hand of them."
- The Thai students liked the "project's wiki" most on the project.
- The US developers affirmed that they "never learn[ed] to use any diagrams such as class diagrams, ERD, activity and sequence diagrams. Diagrams make programming much easier and allow you for a clear view of what you need to implement."
- The coaches and auditors had converging opinions and mentioned that there were: "Too many places to check for communication. It became hard to get oriented with the whole environment" and "to find needed information. Sometimes things slipped by because of this." They also added that "Emails and the wiki provided a massive confusing place to communicate and it was hard to have traceability in [their] communication as a result".

5. LESSONS

Much has been written about the particular challenges associated with distributed GSD (see [6-11] for example coverage). A number of educators have since risen to the challenge of creating a GSD environment and experience to expose their students to some of these realities and to seek ways to overcome the challenges (see [12-16] for a partial set of examples). Concurrently, discussion about suitable tooling environments to support these emerging ways of working has been evolving (see [17-20]).

Ideally, when getting students to work on a software development project for the first time you want to provide a tooling environment that is ready to use and that students can simply start working with. In addition, it needs to align with their learning to date and account for the available technology and communications infrastructure. When this environment needs to account for disparate sets of students, all these factors are unlikely to align, so common ground and compromises are necessary. We have evolved our set of tools organically and the hard way. We therefore consolidate some of our lessons into a table of do's and don'ts to assist other teams of global educators who may be considering setting up a similar initiative for their students in the future. This initial guide is provided in Table 6.

6. RECOMMENDATIONS

This is a paper about student and instructor experiences with GSD projects and, in particular, about the choices to be made about tooling up such complex projects. While our experiences are intended as a repository of knowledge for other educators pursuing this topic, we suggest that there are broader and more general recommendations for industry that arise from our experiences. These are listed below:

1. **Time-zones** - when are the reasonable overlaps so that you can schedule live discussions? Asynchronous communication alone is a sure-fire path to failure. However, alternate so that one side does not always get the short straw when communicating synchronously or the synchronous communication will not be sustainable.
2. **Synchronicity** - some organizations block instant messaging (chats), so make sure you find out what instant messaging clients will work through your firewalls and are in use at the other end. Be sure to pick a system that allows multi-way conversations. Synchronicity is critical at key junctures and technology needs to be there to support it.
3. **Network speed** - what speed do you have or what can you get? What is average and what is the spread? The more bandwidth and reliability of this bandwidth, the more you can do. In our project, the Cambodian team was seriously

impeded in its efforts due to issues in this area that we should have foreseen and accommodated.

4. **Everyday awareness of simple things** - it sounds obvious, but make sure you share a list of people, their job titles, vacations, email and preferred contact mode. Include an organization chart for the project with photos. If you need to escalate a problem, you need to know the hierarchy. Photos help to make people come alive, and professors, auditors and coaches were well known in the more successful teams in our project. The wiki served us well for raising awareness on our global project.
5. **Collaboration / coordination tools** - try and use some form of web-based collaboration tool (e.g., a wiki). This allowed people to share document versions across geographies and acted as the coordination backbone for our project.
6. **I see what you see** – invest in desktop sharing software for requirements, design and code reviews and permit real-time collaborative working. The delay and ambiguity incurred without this technology cannot be understated.
7. **Time to learn other development tools** - requirements management, bug tracking - what have you got? Do you have a preference or will you use the offshore vendor's choices? It takes time to sell the value of a new tool to a team, so factor this into the schedule.
8. **Software versions** - major problems arising from incompatibility between different versions of software should be preventable (e.g., Office 2000/2003/2007). Check this out for all participating sites a priori. We faced unnecessary issues of compatibility of requirements documents across locations, which led to delays and frustration.
9. **Virtual social gatherings** - know when people's birthdays are and hold virtual parties. At our end of project Second Life party many participants came together for the first time, an experience most suggested would have helped establish bonds if undertaken earlier. If it is prohibitive for a team to meet in person, find an alternative way.
10. **Have a trusted cornerstone to the project - visit your offshore team** – it is always appreciated when someone makes the effort to visit. Take time to get to know the people, explain the context of the project, as well as the detail on the project. Our project would not have been viable without the country visits by instructors and trusted relationships that were established before the onset of the student collaborations.

7. CONCLUSIONS

When an organization embarks upon a GSD project for the first time, the primary focus can be upon the benefits to be accrued through cost savings and round-the-clock working. Infrastructure tends to be discussed once prior decisions have been made. Obvious questions arise as to the state of the infrastructure in different locations, local tools used for development, communications and management, and how to blend these preferences seamlessly. Indeed, this very trajectory is one that educators also face. Rather than deal with such delicate alignment issues, it may be common practice to simply adopt the entire tooling environment of the vendor (or in an educational setting, that of the instigator) -- a decision that is not without problems if taken blindly. Educational experiences can expose many of the issues likely to challenge a commercial GSD arrangement and that are in need of careful prior consideration.

Table 6. Tooling do's and don'ts for educators

Do	Don't
Assess the experience and level of exposure of the students to the Internet, as well as to engineering, communication, project management and socialization tooling, and account for the learning curve in the schedule.	Underestimate the importance of communication, project management and socialization tooling by only giving attention to engineering tools; socialization has to have a central place and be sustained throughout the project.
Explain the rationale behind the choice of each tool and setup to students.	Discard Internet access difficulties at different locations and the different perceptions on the usefulness of various tools.
Create a shared consensual tooling environment across locations that reflects what is used in industry and accommodates the mix of roles, cultures and levels of prior exposure to the Internet and technologies.	Overestimate the extent of the use of the tooling by the students; they use less than 50% of their functionality and have to be guided along the way to use them appropriately.
Provide customized training prior to using the tools online or / and on-site – complemented by tutorials and access to resources.	Rely on too many tools and think that tools will solve all the encountered problems; people and processes are essential behind the tools.
Get multiple and redundant communication channels privileging one-to-many rather than one-to-one communications. Wikis also work best in conjunction with short status meetings via chats or conference calls.	Rely on having students install the tooling; the tooling environment must be ready to be used by the students from day one, and set up by the instructors or preferably the IT department.

8. ACKNOWLEDGEMENTS

This work is supported by a National Collegiate Inventors and Innovators Alliance grant (#3465-06), "Incubating the Next Generation of GSD Entrepreneurs" and a Campus Second Life scholarship. We thank all 159 students who have been involved in this global initiative to date.

9. REFERENCES

[1] Gotel, O., Scharff, C. and Seng, S. "Preparing Computer Science Students for Global Software Development". In *Proceedings of the IEEE Annual Frontiers in Education Conference, Borders: International, Social & Cultural (FIE 2006),* San Diego, California, USA, 2006.

[2] Gotel, O., Kulkarni, V., Neak, L., Scharff, C. and Seng, S. "Introducing Global Supply Chains into Software Engineering Education". In *Proceedings of the International Conference on Software Engineering Approaches For*

Offshore and Outsourced Development (SEAFOOD 2007), Zurich, Switzerland, 2007.

[3] Gotel, O., Kulkarni, V., Neak, L. and Scharff, C. "Working Across Borders: Overcoming Culturally-Based Technology Challenges in Student Global Software Development". In *Proceedings of the Conference on Software Engineering Education and Training (CSEE&T 2008)*, Charleston, South Carolina, USA, 2008.

[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". In *Proceedings of the International Conference on Software Engineering Approaches For Offshore and Outsourced Development (SEAFOOD 2008)*, Zurich, Switzerland, 2008.

[5] Gotel, O., Kulkarni, V., Neak, L. and Scharff, C. "Integration Starts on Day One in Global Software Development Projects". In *Proceedings of the International Conference on Global Software Engineering (ICGSE 2008)*, Bangalore, India, 2008.

[6] Aspray, W., Mayadas, F. and Vardi, M.Y. *"Globalization and Offshoring of Software"*. A Report of the ACM Job Migration Task Force, 2006.

[7] Herbsleb, J. D. "Global Software Engineering: The Future of Socio-technical Coordination". In *Proceedings of the 29th International Conference on Software Engineering – The Future of Software Engineering (ICSE-FASE 2007)*, Minneapolis, Minnesota, USA, 20-26 May, 2007.

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

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

[10] Olson, J. S. and Olson, G. M. "Culture Surprises in Remote Software Development Teams". In *ACM Queue*, 1(9), December/January, 2003-2004.

[11] Sarker, S., and Sahay, S. "Implications of Space and Time for Distributed Work: An Interpretive Study of US-Norwegian Systems Development Teams". In *European Journal of Information Systems*, Vol. 13, No. 1, pp.3-20, 2004.

[12] Damian, D., Hadwin, A. and Al-Ani, B. "Instructional Design and Assessment Strategies for Teaching Global Software Development: A Framework". In *Proceedings of the 28th International Conference on Software Engineering (ICSE 2006)*, Shanghai, China, May 20-28, 2006.

[13] Hawthorne, M. J. and Perry, D. E. "Software Engineering Education in the Era of Outsourcing, Distributed Development Distributed Development, and Open Source Software: Challenges and Opportunities". In *Proceedings of the 27th International Conference on Software Engineering (ICSE 2005)*, St. Louis, Missouri, USA, May 15-21, 2005.

[14] Petkovic, D., Thompson, G. and Todtenhoefer, R. "Teaching Practical Software Engineering and Global Software Engineering: Evaluation and Comparison. In *Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE'06)*, Bologna, Italy, June 26-28, 2006.

[15] Richardson, I., Milewski, E., Keil, P. and Mullick, N. "Distributed Development – an Education Perspective on the Global Studio Project". In *Proceedings of the 28th International Conference on Software Engineering*, Shanghai, China, 20-28 May, 2006, pp.679–684.

[16] Williams, J.C., Bair, B., Borstler, J., Lethbridge, T.C. and Surendran, K. "Client sponsored projects in software engineering courses". In *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2003),* Reno, Nevada, USA, February 19-23, 2003, ACM Press, New York, NY, pp.401-402.

[17] Aloi, M. and Fortin, W. "Utilizing IBM Rational Tools to Successfully Outsource in a Globally Distributed Development Environment". In *IBM Rational Software Development Conference*, 2007.

[18] Chen, L., de Souza, C.R.B., Hupfer, S., Patterson, J. and Ross, S. "Building Collaboration into IDEs". In *ACM Queue*, 1(9), December / January, 2003-2004.

[19] Hawthorne, M.J. and Perry, D.E. "Software Engineering Education in the Era of Outsourcing, Distributed Development Distributed Development, and Open Source Software: Challenges and Opportunities". In *Proceedings of the 27th International Conference on Software Engineering (ICSE 2005)*, St. Louis, Missouri, USA, 15-21 May, 2005, pp.643–644.

[20] Mawdsley. J. "Tools for Distributed Development". In *Dr. Dobbs Journal*, September 26, 2007.