CHALLENGES IN TEACHING GLOBAL SOFTWARE ENGINEERING TO UNDERGRADUATE STUDENTS: COURSE DESIGN

Vinitha Hannah Subburaj
West Texas A&M University, School of Engineering, Computer Science, Mathematics, Texas-USA
vsubburaj@wtamu.edu

Emily M. Hunt
West Texas A&M University, School of Engineering, Computer Science, Mathematics, Texas-USA
ehunt@wtamu.edu

Angela Spaulding
West Texas A&M University, Killgore Research Center, Texas-USA
aspaulding@wtamu.edu

James D. Webb
West Texas A&M University, IoT Innovation Laboratory, Texas-USA
jwebb@wtamu.edu

Abstract: Unlike many courses in the field of computer science, teaching software engineering comes with a set of challenges. These major challenges can be categorized into five aspects, namely: (1) incorporating a case study based approach to the design of lectures and assignments, (2) including projects from a range of domains, technology, and platforms, (3) keeping up with rapid evolution of technology, (4) setting up a development environment enabling students to understand the impact of geographical, social, and cultural implications on software development, and (5) having students understand the fact that software development is not simply a technical activity, but involves facilitating effective operation of teams. Since software systems have now become an integral part of almost every single industry, producing students who can develop and maintain systems that span across various industries is critical. This paper describes each of these challenges and possible approaches towards overcoming these challenges. The focus of this paper will address the challenges of creating a course within an undergraduate computer science curriculum to teach global software engineering. Due to the globalization of software development activities, industries are looking at recruiting students who are equipped with skills needed to deal with challenges around global software engineering. Designing instructional materials and assessment tools to develop this unique mix of skill sets is addressed in this paper. We also discuss both the traditional and non-traditional aspects of teaching software engineering to computer science students.

Keywords: Global software engineering, Course design, Challenges

Introduction

Software engineering has become an important course in the CS curriculum. Students entering into software engineering companies are expected to have a wide range of skills sets ranging from communication to purely technical skills. Most importantly, today's computing industries operate across the globe where either their customers are from a different country or their development team operates from another country. Industries hiring CS students have a general observation that our students struggle with projects that require them to be operating in a global environment. Organizations have started to research and invest on bringing Global Software Engineering (GSE) into classrooms and teach students the needed skill set enabling them to succeed in a global project development environment. This paper is also one such effort of teaching GSE to undergraduate CS students and add it as a required course into the CS curriculum.

Global Software Engineering involves software engineering practices carried over a global setting. Today's software development activities get distributed across the globe mainly to lower the development cost. By outsourcing development tasks to low-wage countries, the overall software development cost can be brought down. This introduces a scenario of global software development team who have to operate together to build a reliable software product. To remain competitive in the market, software companies have to deliver a product that gets used by global customers. This furthermore adds on challenges of building a software product that should satisfy requirements from multi-cultural settings, varying political, ethical, and societal backgrounds. Global software development teams have to ensure proper communication and planning to overcome language and time zone barriers. Non – functional aspects of the software like look and feel, ease of use, and aesthetics vary across the globe. If a software is built for global customers, then care should be taken to ensure that these non-functional aspects get addressed during the software design.

Teaching software engineering to Undergraduate students has become challenging due to the complex nature of evolving software systems. Simulating real world software development inside a classroom setting is not easy due to several resource constraints. Educators have been working on pedagogies that can effectively simulate the real-world software development experience for students. Teaching global software engineering is even more challenging compared to teaching software engineering. Simulating a global software engineering environment to teach the principles and practices of global software engineering is very challenging. In this paper, a sample syllabus is presented for educators to use in their curriculum. The global software engineering course design proposed in this paper has been done after careful analysis and through the experience of many years of offering software engineering coursework.

The rest of this paper is organized with a related work section, a teaching software engineering section, a global software engineering – course design section, and a summary section.

Related Work

Urban [11] addressed software engineering on the web through a graduate level course on software requirements and specifications. The course project involved the development of a web-based software tool that implemented the ANSI/IEEE standard on software requirements specifications. The graduate course was offered during the Fall 1998 semester. The concepts developed in the graduate course were followed through into an undergraduate senior-level software engineering project two course sequence during 1999. There were several other IEEE software engineering standards developed into web-based software engineering tools during subsequent years.

Deiters, Constanze, et al. [4] stated that depending on the stakeholder located in a particular region, the distribution of the software development projects varies. There must always be a good combination of theory and practice for a software engineering program in a university. The paper brings forth the concepts behind the common teaching atmosphere for global software engineering called the GlobSELab. Based on the feedbacks obtained from the participants, the GlobSELab was added to the course. The lab describes the teaching intentions, project management, and quality assurance. Motivated by the experiences of the distributed practical course, foreign universities were invited to create a platform that correlate more of an industrial reality. The course will be of great benefit to provide solutions to typical problems of global software engineering.

Paasivaara, Maria, et al. [10] developed a course where they incorporated the concepts of distributed Scrum in a global software engineering (GSE) environment. The course adapts a combination of both agile methodologies and industry best practices. The previously used GSE courses used plan-driven methods. Whereas the distributed Scrum method is unique in assessing the student expectations and learning. A mixed-method approach has been employed to assess the learning, distributed collaboration, building trust, and inter-cultural collaboration. The results obtained from the analysis of data before, during, and after the course yielded the discussion about the

challenges in applying the skills, strategies to overcome the challenges, and the strategy effectiveness. Hence, distributed Scrum in combination with GSE would be considered as an important course design.

Lescher, Christian, et al. [6] in their paper have stated that GSE has brought forth several new challenges in the market. Some of them are geographic separation, various time zones, cultural and language barriers which causes a delay in the communication and often leads to quality and cost issues. In the paper, GSE was taught in two different classrooms in order to compare them with the traditional classroom setting. This approach resulted in the reflection on GSE key effects on communication issues, and distribution delay. This effort is intended to expand the work on interactive GSE exercises and to extend the set of exercises by evaluating additional exercises for teaching GSE.

Nordio, Martin, et al. [9] discussed several challenges that a software engineering student will face in distributed software development. A software engineering course was taught using globally distributed projects with an aim to prepare the student to meet such challenges. The paper presents the experience regarding an approach used to teach distributed software engineering. Even though the approach is an old method, improvements have been made based on the lessons learned by the authors. The API design has an important role in this approach. In addition, this approach has an emphasis on the development of communication skills as almost 30% of the time is spent by the student in a project by corresponding to communication.

Beecham, Sarah, et al. [1] adopted a Global Teaming Model framework to describe the requirements of global software development. From the assessment of three small or medium sized enterprises (SME), GTM practices that are relevant to SMEs have been identified. Assessment was also done on the gap between practices addressed by GSE-Ed literature and the needs of SMEs engaged in GSD. Seven GTM practices were relevant and two were lacking. The analysis brings forth the complexity involved in the roles and responsibilities of the instructors in GSE-Ed courses. Hence, students face the reality that practitioners of SMEs need to actively participate in the education process. Beecham, Sarah, et al. [2] have also conducted a study in offering different options to CS educators teaching CS courses in a global setting. They specifically focussed on learning GSE theory and learning GSE by doing. Studies that take a hybrid approach of combining theory and practice were also included in this paper.

Matthes, Florian, et al. [8] stated that international aspects must be included in the education of software engineers along with technology and management. The paper described an applied approach that involved 43 participants at 5 different distributed academic institutions. The paper presented the lessons learned from recommendations by teaching staff and students. The approach introduced is expected to serve as a base foundation for similar GSE ventures.

Li, Yang, et al. [7] stated that software engineering is now facing challenges due to globalization. Many industries ensure global competitiveness by transferring a part of their development activities to distributed countries. Instructors face the problem of incorporating skills related to recent developments in global software engineering. The paper describes the exercises required for teaching GSE in a single class room and report the experiences. The students gain experience to work with various time zones and time management. Hence, such exercises could be included in the course curricula.

Kuhrmann, Marco, and Jürgen Münch [5] stated the importance to understand the need for interdisciplinary teamwork for a successful project execution. A course unit discussed in this paper, was used to create an awareness among students regarding the role of communication in distributed software development. The course unit presents: 1) an environment in which students can learn distributed agile software and 2) a controlled experiment instrument for organizing a small software project to be carried out in virtual teams. Some of the challenges faced by the students are to overcome the limitations, set up teams, and develop the application. The results due to poorly organized communication indicated that there were issues regarding technical, architecture and developmental resources. Hence, the lack of communication protocols can impact the team's coordination and performance.

Damian, Daniela, et al. [3] presented a framework for teaching GSD skills in collaboration with three universities. The findings from their research show that the students learned to recognize the importance of effective communication between clients and developers and how the GSD environment influenced the communication.

Teaching software engineering

The following are the issues and some solutions on addressing those issues while teaching software engineering.

(1) incorporating a case study based approach to the design of lectures and assignments,

Adopting a single case study throughout the course work has proven not to be successful while covering different concepts of software engineering. While the other extreme of using too many case studies inside the coursework have also often confused the students. The approach that has proven to be useful is usually to take two different case study examples for preparing the lecture materials and use two or more complete different ones for coming up with assignments and in-class activities. Through this approach, the students get to understand the concepts by applying them to just enough case study examples. Usually referring to more than one textbook for case studies and also looking out for more real-world examples to help them understand the real-life scenarios has proved to be very successful.

(2) including projects from a range of domains, technology, and platforms,

Selecting appropriate projects from a range of domains, technology, and platforms is a key factor in teaching software engineering. By including projects from different domains, we can prepare students with skills sets needed to be successful in the real-world. A real challenge is developing these projects and preparing the students with the background information needed. For instance, if we include software projects from the healthcare domain, how do we prepare students with the background information on healthcare industries in general to help them build projects in this domain? How much time do we spend investigating the domain before we let the students perform the actual development? One of the solution to this problem is to have a system analysis and design course as a pre-requisite to this course where we can focus on teaching students analysis and design techniques. Requirements analysis techniques, such as root-cause analysis, informal benchmarking, observation, and outcome analysis, can be used to investigate the problems that come from different domains.

(3) keeping up with rapid evolution of technology,

The projects of choice and the technology used during software development should reflect current technology. By keeping up with the technology, we not only teach students on how to build with latest technology, but also use the latest CASE tools and techniques to aid project development. This situation is crucial and challenging due to rapid growth of software technology. By letting the students do technical feasibility analysis during the initial stages of project development, incorporating the use of CASE tools during software development, and the choice of hardware and software based on the industry needs will help handle this situation.

(4) setting up a development environment for enabling students to understand the impact of geographical, social, and cultural implications on software development, and

This challenging aspect addresses problems beyond just solving the problem and implementing a solution. Students usually try to overlook this phase and underestimate the importance. The non-functional aspects of software development needs to be included during the early phases of software development. Conducting a pre and post mortem analysis with respect to these aspects during software project development will help students understand and experiment the impact of geographical, social, and cultural implications during software development.

(5) having students understand the fact that software development is not simply a technical activity, but involves facilitating effective operation of teams.

Teamwork is an essential skill required for software engineering jobs. Efficient team management skills and their effective operation is key to successful completion of the project. Students come in with varying interoperable skill sets and are required to learn the importance of being responsible team players. Some of the proven methods of helping students to work efficiently in teams include: effective team formation, practice of recording team meeting minutes, maintaining an anonymous online team resolution center where students can report team problems and get solutions, having 2-3 teamwork assessments done during the course of project development, and having a percentage of the grade assigned for effectively working in teams.

GSE – Course Design

The prerequisite of this course should be a Requirements Engineering or System Analysis and Design course. Students should be able to clearly state the non-functional aspects of the project, such as the geographical, social, and cultural implications as requirements. Course design of a GSE course is provided in Appendix A. The course description, objectives, and the major components of the course are highlighted in the syllabus. 50% of the total weight has been assigned to global project development and the components with descriptions are detailed in Table 1.

This course provides the students with an opportunity that in the GSE course is unique when considering that many other course for group projects puts together students who immediately begin development of the software from a problem statement. The team formation activities that occur early in the GSE course will set the stage for enhanced communication, member strength analysis, and project management aspects.

The concepts in the GSE course adds a level of complexity to a software engineering course project that is not experienced in most other software engineering projects. The success of the students will transfer into software engineers who are industry ready.

Summary

Current literature clearly supports the need for global software engineering courses at the undergraduate level. This paper has provided the motivation and survey of earlier efforts for the development of global software engineering courses in a CS curriculum which includes issues and relevant solutions in teaching software engineering. The authors present GSE – Course Design which elaborated on one instance of a GSE course at the course information level including the syllabus that will be used during implementation fall 2018. The next step will be to gather data on the further design and implementation of the stated GSE course. Through continuous process improvement, the constituents will help drive course enhancements.

Appendix A
CS – XXXX - Global Software Engineering

Prerequisites
Requirements Engineering or System Analysis and Design
Course Descriptions
This course teaches the essential skills necessary to develop software systems in a global environment. This course will cover fundamental topics of a global software engineering life cycle process from requirements specifications to testing of a completed software system in a global setting. The course has an emphasis on essential communication skills required by the students to effectively conduct the software development process in a global setting. This course is project based involving practical implications along with team work. Projects for this course will be approved by the instructor in advance and will be originating from different countries. A major part of the course will involve students to work with global stakeholders to design and develop software systems. Students will be supervised and are expected to be well organized while working with team members and developing their presentation and management skills.

Objectives
After completion of the course, students will be able to

- Select appropriate software life cycle process models to be used for global software development
- Describe and apply fundamentals of software engineering methodologies and techniques to build projects on a global setting
- Recognize the importance and challenging aspects of gathering software requirements especially when the customers are geographically distant
- Translate software requirements to design, design to code, and then test the software system based on appropriate global software engineering methodologies
- Choose appropriate CASE tools, models, design patterns, architecture, and programming language for global software engineering
- Employ team work – that includes project management skills, interpersonal, and communication skills in a global setting
- Describe different software testing methodologies that have been effective with global software engineering

Grading Policy

Class attendance and participation	5%
Homework/Lab assignments	15%
Project	50%
Exam (2 midterms 10% each, 1 final 10%) :	30%
Total:	100%

Table 1: Project Grade Distribution

Here is the breakdown of how the project grade is being calculated.

S.No	Tasks	Description	Weights
1.	Project plan and feasibility analysis	During the first two weeks, students are required to establish communication with their global customer as directed by the instructor. They are required to come up with a project plan and also conduct feasibility study.	5%
2.	Decide on a software life cycle model and establish modes of contact with the global customer	Choice of a suitable software development life cycle (SDLC) model along with effective modes of communication between the team and customer is very important.	5%
3.	Requirements analysis	Based on the choice of SDLC, this phase would differ. A good understanding of the requirements is essential for all project development.	10%
4.	Design	Design and implementation will be done by the project teams with constant feedback from their global customer.	15%
5.	Implementation		20%
6.	Testing	Testing will be done by another team different from the ones who developed this project. Test data and the methodology should be clearly specified by the project developers. The other team should consider themselves working in a different country and follow the practices followed in that specific country to conduct the testing.	10%
7.	Appropriate use of software tools Post-mortem analysis of what went well and what can be improved	The appropriate use of technology will be weighed as a factor contributing to the performance of the project members. Factors that led to success and failure of the project will be documented by the student.	10%
8.	Documentation Meeting minutes	Documentation of the entire project development along with the minutes recorded during every meeting should be reported.	10%
9.	Customer evaluation and feedback	Rubrics will be provided to the customers for evaluating the teams working on their requirements.	5%
10.	Final Presentation		10%
	Total:		**100%**

References

Beecham, Sarah, et al. "Challenges and recommendations for the design and conduct of global software engineering courses: A systematic review protocol." Tech. Rport Lero_TR_2015_01 (2015).

Beecham, Sarah, Tony Clear, and John Noll. "Do we teach the right thing?: A comparison of global software engineering education and practice." Proceedings of the 12th International Conference on Global Software Engineering. IEEE Press, 2017.

Damian, Daniela, Allyson Hadwin, and Ban Al-Ani. "Instructional design and assessment strategies for teaching global software development: A framework." Proceedings of the 28th International Conference on Software engineering. ACM, 2006.

Deiters, Constanze, et al. "Glose-lab: Teaching global software engineering." Global Software Engineering (ICGSE), 2011 6th IEEE International Conference on. IEEE, 2011.

Kuhrmann, Marco, and Jürgen Münch. "Distributed software development with one hand tied behind the back: A course unit to experience the role of communication in GSD." Global Software Engineering Workshops (ICGSEW), 2016 IEEE 11th International Conference on. IEEE, 2016.

Lescher, Christian, Yang Li, and Bernd Bruegge. "Teaching global software engineering: Interactive exercises for the classroom." Global Software Engineering (ICGSE), 2014 IEEE 9th International Conference on. IEEE, 2014.

Li, Yang, et al. "Teaching global software engineering by simulating a global project in the classroom." Proceedings of the 47th ACM Technical Symposium on Computing Science Education. ACM, 2016.

Matthes, Florian, et al. "Teaching global software engineering and international project management." Proceedings of the 3rd International Conference on Computer Supported Education. Noordwijkerhout, Netherlands. 2011.

Nordio, Martin, et al. "Teaching software engineering using globally distributed projects: The DOSE course." Proceedings of the 2011 Community Building Workshop on Collaborative Teaching of Globally Distributed Software Development. ACM, 2011.

Paasivaara, Maria, et al. "Teaching students global software engineering skills using distributed scrum." Software Engineering (ICSE), 2013 35th International Conference on. IEEE, 2013.

Urban, Joseph. “Software engineering on the web.” Proceedings of the 3rdInternational Conference on Business Information Systems, Springer, 1999.

