Exploring and Expanding GSE Education with Open Source Software Development

RUNE HJELSVOLD and DEEPTI MISHRA, The Norwegian University of Science and Technology, Department of Computer Science

Global software engineering (GSE) courses traditionally require cooperation between at least two universities so as to provide a distributed development environment to the students. In this study, we explore an alternative way to organize a global software engineering course where students work on open source software development (OSSD) projects rather than in a multi-university collaboration setting. The results show that the new setup may provide core GSE challenges as well as challenges associated with software development outsourcing and challenges related to working on large open source software. The present article compares the experiences gained from running a combined GSE and OSSD course against the experiences gained from running a traditional GSE course. The two alternatives are compared in terms of students' learning outcomes and course organization. The authors found that a combined GSE and OSSD course provides learning opportunities that are partly overlapping with, and partly complementary to, a traditional GSE course. The authors also found that the combined OSSD and GSE course was somewhat easier to organize because most of the activities took place in a single university setting. The authors used the extended GSE taxonomy for the comparison and found it to be a useful tool for this, although it had some limitations in expressive power. Therefore, two additional relationship dimensions are proposed that will further enrich the extended taxonomy in classifying GSE (and OSSD) projects.

CCS Concepts: • **Software and its engineering → Software development process management**; **Agile software development**; *Open source model*; *Programming teams*;

Additional Key Words and Phrases: Global software engineering, GSE Education, GSE Taxonomy, Open Source Software Development, Software Development Outsourcing, OSS

ACM Reference format:
Rune Hjelsvold and Deepti Mishra. 2019. Exploring and Expanding GSE Education with Open Source Software Development. *ACM Trans. Comput. Educ.* 19, 2, Article 12 (January 2019), 23 pages.
https://doi.org/10.1145/3230012

12

1 INTRODUCTION

Software engineering students are required to attain more than just content knowledge. Their learning should include mastering skills and attitudes for developing software for real world usage, which is difficult to achieve inside classrooms alone. Global competition, the need for flexibility, the need for new types of expertise, and the struggle for cost efficiency, drives software

This study was partly funded by Excited—the Centre for Excellent IT Education (https://www.ntnu.edu/excited).
Authors' addresses: R. Hjelsvold and D. Mishra, The Norwegian University of Science and Technology, Department of Computer Science, Teknologiveien 22, 2815 Gjøvik, Norway; emails: {rune.hjelsvold, deepti.mishra}@ntnu.no.

1946-6226/2019/01-ART12 $15.00
https://doi.org/10.1145/3230012

development companies to engage in distributed software projects (Lacity et al. 2009; Persson et al. 2009; Mishra and Mishra 2010, 2011, 2012). Therefore, the skill set that the learner attains must be what the software industry currently requires (Mishra and Yazici 2011). As a result, global software engineering has emerged as a part of the academic curriculum in computer science degrees. Academia is trying to simulate real world experiences by having students work in cross-university, distributed teams while developing software. These projects may have real or fake customers.

In the paper "How Best to Teach Global Software Engineering? Educators are Divided." (Beecham et al. 2017), Sarah Beecham interviews five different educators. These interviews show that educators are divided about the best way to organize a global software engineering (GSE) course. Some educators consider it important that such courses include global virtual collaborations in multisite, cross-university settings. Others suggest that students could be introduced to GSE through the participation in global open source projects.

A course on global software engineering has been offered at the Gjøvik campus of our university since 2013. The course was the result of an international collaboration with the Tampere University of Technology, Finland, until 2015, allowing for geographically distributed teams. The collaboration came to an end in 2016, however, due to lack of student interest at the partner university. As a consequence, our university decided to explore the open source software [OSS] development (OSSD) alternative.

The MOODLE project[1] was chosen as the basis for the OSSD alternative for the course. In MOODLE's governance model, the MOODLE headquarters (HQ) is responsible for maintaining the software. Therefore, the students experienced working in a setting that resembles a software development outsourcing setting: the student teams acted as distant teams choosing tasks defined by the headquarters, implementing them according to the standards and procedures defined by the headquarters, and submitting their code to the headquarters for code review and eventual release.

In this article, we will share our experiences from exploring the OSSD alternative. We will present and discuss the innovative parts of the course. We will also make a systematic comparison, although based on small sample sets, of this new way of running the course with the former, multi-university, distributed team approach (the traditional GSE education approach). The paper is based on a rather small case study but we still consider it a useful contribution to an interesting, but still unanswered, question in the GSE education community.

1.1 Motivation and Research Questions

The main motivation for the article is to study whether OSSD is a viable alternative for GSE education. This study aims to explore the following research questions:

- How well can core GSE issues be covered when organizing GSE education projects as OSSD projects, and to what degree may research literature be used to fill possibly missing parts?
- What software outsourcing experience may students get from participating in OSSD projects?
- What additional global issues may students experience from participating in OSSD?
- What are the additional challenges for the teaching staff when using OSSD rather than the traditional multi-university approach?

1.2 Article Organization

This article is organized as follows: we will first present a short review of papers related to GSE and OSSD in general followed by a short review of papers discussing the use of OSSD in GSE

[1] https://MOODLE.com/.

specifically. We will then describe how this study was set up and the empirical data we collected through the study. Based on these data, we compare the use of OSSD in GSE education with the more common approach of having distributed software development teams. A summary of the lessons learned and some ideas for future work conclude the paper.

2 LITERATURE REVIEW

In their systematic review, Clear et al. (2015) adopted the following definition for GSE/Global Software Development (GSD):

> *In GSD, stakeholders from different national and organizational cultures and time zones are involved in developing software … and tasks at various stages of the software lifecycle may be separated and implemented at different geographic locations coordinated through the use of information and communication technologies …*

As mentioned before, GSE educators are divided in their views on how to best run GSE courses. Sarah Beecham interviewed five different educators about how best to teach global software engineering (GSE). In these interviews (Beecham et al. 2017), Clear and Damian emphasized the need for such courses to include global virtual collaborations in multisite, cross-university settings. However, they also cautioned that such an educational environment will require additional effort, strategic work, and instructor resilience when it comes to setting up, teaching, and assessing student work.

In the same interview, Barr highlighted several challenges of conducting a GSE course in cross-university settings, such as course design and organization; class management; sustainability, scalability and reusability; students' different work ethics and skill levels; and different communication and development tools in different universities. He proposed the participation in global open source projects as an alternative way to introduce students to GSE since open source software projects generally are globally distributed projects. Says Associate Professor John Barr:

> *If the OSS project is indeed global, students are introduced to GSE challenges and can participate in the GSE process—GSE education's major goals.* (Beecham et al. 2017).

An open source project is not the same as a GSE project in a commercial company. It may still, however, meet core GSE education's learning objectives (Beecham et al. 2017). Running GSE courses as OSSD projects give some additional benefits:

- It may require less organizational and teaching efforts (Beecham et al. 2017).
- An open source project is a real project for a solution that is often in widespread use and therefore it is highly motivating (Beecham et al. 2017).
- It mitigates many GSE educational constraints and reduces many institutional challenges (Beecham et al. 2017).
- Developers on OSSD projects usually have to get to know a potentially large legacy code base (Smith et al. 2014; Fagerholm et al. 2013; among others).

2.1 GSE Education in Cross-University, Distributed Teams

Clear et al. (2015) asserted that, to be considered GSE education, software engineering education must address the notions of distance common in the GSE literature:

- Geographic distance
- Temporal distance
- Cultural distance (which they further divided into linguistic and institutional distances)

The 82 papers reviewed by Clear et al. (2015) appeared to report on projects where two or more universities collaborate on a rather equal basis and where each development team acts as an autonomous unit.

In their paper, Šmite et al. (2014) presented a taxonomy for GSE built on the same notions of distance. This taxonomy defines a legal entity as a special type of institutional distance element concerned with whether the development is conducted within the same company (insourcing) or with another company (outsourcing). The review conducted by Clear et al. (2015) did not include outsourcing when coding the reviewed papers, thereby indicating that outsourcing is not a topic typically covered in GSE education.

Britto et al. (2016) have suggested an extension to the Šmite taxonomy that adds a "setting" dimension further divided into four all new on-site dimensions describing the characteristics of the individual GSE sites and two all new relationship dimensions. The taxonomy is intended to make it easier to analyze and compare results from different GSE studies.

The systematic review conducted by Clear et al. (2015) identified 45 top recommendations to GSE educators. Among these are to use self-organizing teams, to encourage a reflective attitude in students, and to require regular status meetings and/or reports. The review also lists agile methods as being recommended by GSE education researchers. Reflection in agile teams is a topic studied in more detail by Babb et al. (2014). They distinguished between reflection-in-action, which is the reflection done by the team members as they solve actual project issues, and reflection-on-action, which is post-action reflection and evaluation of the teamwork. They proposed a Reflective Agile Learning Model (REALM) where they suggested reflective and learning opportunities related to agile practices.

We have followed the recommendations of using agile methods, self-organizing, and reflecting teams in our project.

2.2 Teaching Outsourcing

Outsourcing is recognized as a significant challenge in the GSE research community:

- Moe et al. (2012) studied three companies that terminated their offshore outsourcing development work to find the main reasons for termination. They found that disappointingly low software quality was the main reason for termination, and attributed this to a large degree to communication and motivation issues.
- Kumar and Thangavelu (2013) also studied factors that impact the quality of code developed through outsourcing and draw the following conclusion:

  > *Our empirical study revealed that offshore and onsite teams knowledge sharing and knowledge transfer has been key determinant to create a significant impact of GSD project outcome.* Kumar and Thangavelu (2013)

- The study conducted by Kannabiran and Sankaran (2011) shows that requirements uncertainty has significant impact on the quality of the code developed through offshoring (offshore outsourcing) while process maturity and trained personnel have moderate impact.

There is not much research present on outsourcing education. Gotel et al. (2007) reported on a global software development project where students from three globally distributed educational institutions collaborated. In their set-up, three teams of US students were tasked to develop different software products for Cambodian clients, while subcontracting the database component to third-party teams of Indian students. The project specifically demonstrated that cultural distance is an issue students also have to handle in outsourcing projects. Clear et al. (2015) noted the value of outsourcing as a strategy to address mixed team skills, by allowing student teams to pay to

outsource parts of their project to a global developer (Long 2010). They further noted this was not without its challenges, for instance, two of the teams had bad attitudes related to the use of an outsource programmer. They were unwilling to bring this third party into the team as a contributing partner (Long 2010).

Honig and Prasad (2007) developed and implemented an outsourcing experience for students in an advanced software engineering course in which students at two universities were organized in teams where each local team would have their own project, but with a team at the other university as the outsourcing centre. One of the main lessons learned in this project is that the students learned to acknowledge the fact that outsourcing requires extra attention and team resources to be successful. While 92% of the students agreed (or agreed strongly) to the statement that outsourcing was primarily a way to save money when asked before the projects started, only 67% shared this view at the end of the project, indicating the students have acquired a deeper and more diverse understanding of the pros and cons of outsourcing.

From the outsourcing literature, we conclude that communication and information-sharing issues are key challenges in software outsourcing, that trained personnel are important for the success of such projects, and that students may acquire a deeper, more diverse understanding of the pros and cons of outsourcing from participating in an outsourcing project.

2.3 Open Source Software Development Education

Finding and choosing the right project for OSSD in software engineering education is challenging (Smith et al. 2014). For GSE it is even more challenging as the projects should give the students the opportunity to experience challenges that are important to the very global aspect of GSE, and not include too many non-GSE related challenges. Large code bases are one aspect of software engineering not specific to global projects and therefore may be considered a non-GSE challenge. Large-coded bases, however, are quite typical for real GSE projects.

Onboarding is a term used to describe how new developers acquire the knowledge, skills, and behaviors needed to become productive. In their study of onboarding in open source software, Fagerholm et al. (2013) described the importance of OSS this way:

> *In many application domains, engagement with OSS is an inevitable part of the computing business. In order to participate in existing software ecosystems (such as Android or the LAMP stack), companies may need to adopt open source development and licensing approaches in order to combine their offering with the infrastructure provided by the ecosystem. Another reason to participate in open source projects is to be able to conduct projects of a size that exceeds the capabilities of an individual organisation.*

The Undergraduate Capstone Open Source Projects (UCOSP) is one of the larger OSSD education programs (Stroulia et al. 2011; Holmes et al. 2014). In UCOSP, teams of five to eight undergraduate students geographically distributed across Canada, and possibly other parts of North America, work together on OSSD projects. By 2014, 400 students had participated in the program over the course of five years (ten terms) according to Holmes et al. (2014). UCOSP has three primary learning objectives (Holmes et al. 2014):

- Giving students experience in working on real distributed OSSD projects as full members of the development teams.
- Giving students the opportunity to integrate and apply the skills they have learned in a real development setting
- Giving students the opportunity to develop and improve their technical communication skills in a real development setting where the majority of the interaction takes place online.

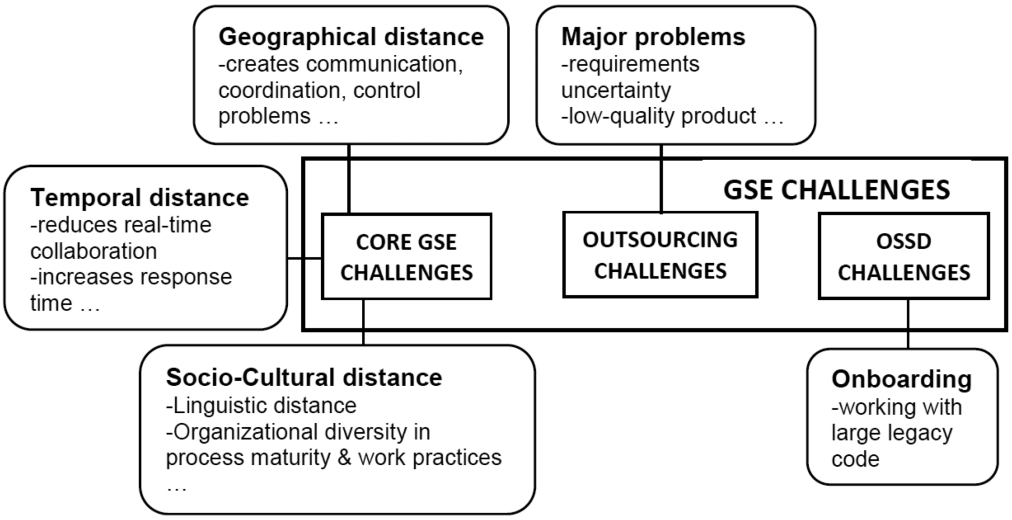

Fig. 1. Challenges related to GSE, OSSD, and outsourcing.

The UCOSP program does not seem to emphasize the GSE aspects of OSSD specifically. The teams are distributed, but the geographic, temporal, and cultural distances are rather limited and do not seem to be addressed explicitly by the educators.

Organizing OSSD educational activities raises some issues that are not seen in typical GSE education. The aforementioned onboarding is one such issue. The UCOSP program recommends using experienced developers from the actual OSSD communities as mentors that will help the teams learn the specifics of the given OSSD project. Other researchers, such as Fagerholm et al. (2013), emphasized that teams should have some freedom to organize their work, within the constraints of the actual OSSD project, and therefore recommend having self-organized teams. Fagerholm et al. (2013) conclude:

> *We thus find support for the hypothesis that onboarding support increases the chance of developers to be exposed to, select, and perform tasks in a proactive and self-directed manner in open source projects. We assume that this applies in other kinds of GSD settings where the organisational and team structure is similar.*

2.4 Revisiting GSE Education in the Context of OSSD

There is not much literature on combining GSE and OSSD in education. It may therefore be useful to revisit some of the fundamentals of GSE education.

GSE Challenges. As pointed out by Clear et al. (2015), GSE education should be focusing on geographic, temporal, and cultural differences. There are, however, global challenges related to outsourcing and there are OSSD challenges that are relevant to GSE and GSE education. Figure 1 shows some of the major challenges related to GSE, outsourcing, and OSSD. In the figure, outsourcing is separated from the core GSE challenges, even though it is one of the dimensions in the extended GSE taxonomy. The reason for this is outsourcing is often included implicitly—and many times only incidentally—as a part of GSE studies, in a form of role-play between teams and sites.

The GSE Distances. The GSE distances mentioned above cannot alone be used to describe the differences between various types of GSE and OSSD projects. The members of a multi-site GSE project working around the clock, for instance, may, according to the Šmite et al. (2014) taxonomy

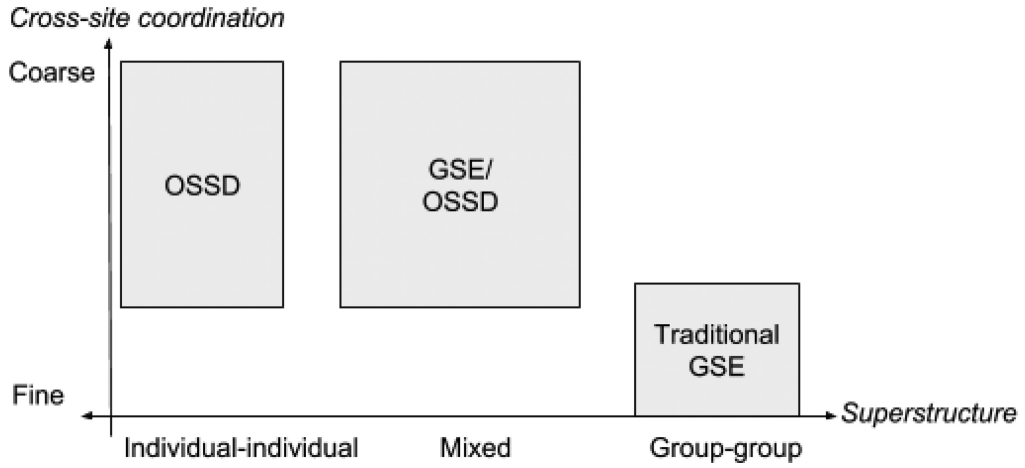

Fig. 2. Comparing different types of projects using the new dimensions.

for GSE, be classified as having *long geographic distance* and *large temporal distance*. An OSSD project may have developers scattered around the world working on a new feature to be included in the future version of the software system. Such a project could also be classified as having *long geographic distance* and *large temporal distance*. There are, however, significant differences in the impact these distances may have on the two projects. Šmite et al. (2014) already discussed how global and temporal differences need to be considered in relation to whether the project is an *onshore* or an *offshore* project.

One project characteristic that may be of use when comparing GSE and OSSD projects is the degree to which site activities need to be coordinated. A typical case for GSE projects, for instance, is the need for fine-grained **cross-site coordination** because activities at the various sites are tightly coupled. A typical case for an OSSD project, on the other hand, is coarse-grained cross-site coordination because of the loose coupling between the individual developers.

Another difference that we may need to consider when comparing OSSD and GSE projects is the overall project structure, the **superstructure**. A typical GSE project, for instance, consists of two or more groups of developers—each group located at one physical site. A typical case for OSSD projects, however, is having developers working individually and not as part of a co-located sub-team. There are GSE and OSSD projects with mixed superstructures too, where some team members are co-located and some work individually. Figure 2 illustrates how GSE, OSSD, and GSE/OSSD project may be classified according to these new dimensions.

Section 5 gives a more in-depth comparison of a GSE and a GSE/OSSD course with respect to the extended GSE taxonomy (Britto et al. 2016) along with proposed additional dimensions.

3 COURSE ORGANIZATION AND DATA COLLECTION

The traditional GSE version of the course was presented and discussed in Sievi-Korte et al. (2015) therefore its details will not be discussed in this article. In this section, we will describe the details of the GSE/OSSD version.

3.1 Course Details

The global software engineering course is a mandatory third semester course in a two-year international master program at our university.

3.2 Participants

Twelve master degree students—seven domestic and five of Eastern European origin—signed up for the GSE/OSSD version of the course. Students must possess a bachelor degree in computer science (CS) to be accepted into the master program. All students were familiar with agile methodologies, Scrum, and the JIRA[2]/Confluence[3]/Git[4] tools from earlier projects in the master program, but none had former experience in GSE or OSSD. The students formed two teams of mixed nationality. Both teams used JIRA as an issue tracker and sprint planner with planning poker functionality in order to manage the Scrum development process. They also used Slack[5] and other tools such as Google Hangouts[6] and Skype[7], for communication among the team members. Students tried to meet face-to-face as much as they could but, due to other course commitments, they often had to conduct online meetings.

3.3 Procedure

Initially, students were introduced to the course and to the MOODLE learning management system. The student teams explored open issues (requirement specifications) announced by the MOODLE HQ. The students were free to choose their own issues, but their choices had to be approved by the educators. The students were recommended to choose issues that did not require deep and advanced MOODLE system knowledge. One of the student teams chose to select issues listed in the MOODLE issue tracker[8] and decided to work on the mobile version of the MOODLE for the following reasons:

- The possibility of high interaction with the MOODLE community.
- The experience of an open and global development environment.
- The product will be useful for the MOODLE community.

The other team preferred to work on a MOODLE plug-in that gave them independence in terms of working on their own with comparatively less interaction with the MOODLE community, but at the same time to being able to contribute in an open-source project.

The teams worked according to the standards/conventions specified by the MOODLE HQ. After internal code review and testing, the teams submitted their code to the MOODLE HQ for acceptance testing.

The teams followed the SCRUM development model with two-week sprints. Students were required to present their progress, the challenges they faced, and their reflections in seminars held at the end of each sprint.

In parallel to the development work, each team studied research papers related to major areas in GSE such as project management issues including risk management, communication and coordination issues, cultural issues in GSE, and the like. These research papers were selected and allocated to the teams by the course lecturers. Table 1 shows the topics and papers assigned to each team. The projects consisted of six sprints in total. Papers were assigned to the students in four of these six sprints.

At the end of the course, there was a final sprint presentation. Additionally, each team delivered a final report summarizing the team's contributions and reflections. Furthermore, each

[2]https://www.atlassian.com/software/jira.
[3]https://www.atlassian.com/software/confluence.
[4]https://git-scm.com/.
[5]https://slack.com/.
[6]https://hangouts.google.com/.
[7]https://www.skype.com/.
[8]https://tracker.moodle.org/.

Table 1. Research Papers Allocated to Teams

Sprint	Research topic	Team 1	Team 2
1	Project and Risk Management	Šmite (2014)* and Casey (2010)*	Shrivastava and Rathod (2015) and Hossain et al. (2009)
2	Software process model for GSE and OSS	Gary et al. (2011) and Lavazza et al. (2010)	Kroll et al. (2014a), Kroll et al. (2014b) and Richardson et al. (2012)*
3	Software quality in GSE and OSS	Tervonen et al. (2013)* and Jabangwe and Šmite (2012)*	Moe et al. (2012)* and Barney et al. (2011)*
5	Cultural issues and competency requirements for GSE	Schlichter and Persson (2014)*, Sonne et al. (2013)* and Søderberg et al. (2013)*	Holtkamp et al. (2015)* and Holtkamp and Pawlowski (2015)

outsourcing also discussed.

team member was required to submit an individual report, summarizing personal experiences and reflections.

3.4 Data Collection and Analysis

The present research is qualitative in nature and, as such, requires the methods pertinent to such data analysis. We refer to Nkwi et al. (2001) for a definition of qualitative research. To be more specific, this work employs the methods of the qualitative thematic analysis presented by Aronson (1994) and described in detail by Guest et al. (2012). This procedure includes getting familiar with the data, generating initial codes, searching for themes, reviewing themes, defining and naming themes, and producing the report (Braun and Clarke 2006).

This work is a case study focused on an exhaustive analysis of an isolated situation as well as a comparison of different cases. According to Randolph (2007), the goals of any given case study are to develop an in-depth understanding of a case, or multiple cases, and to gain insight into the interaction between the phenomenon and the case. In a case study, the researchers use several sources of evidence, such as archival records, direct observations, interviews, and documents. Then they analyze that data through pattern-matching and explanation-building, and through addressing rival interpretations.

In the present study, data is collected from the students' final reports as well as material collected during the seminars held after each sprint. The collected material mostly consists of video recordings from the seminar, and slides and other presentation material used by the students in their presentations.

This research was designed as an inductive thematic analysis—described in Chapter 1 in Guest et al. (2012)—where the main category was initially defined as GSE challenges with sub-categories of geographic, temporal, and socio-cultural distance. However, while examining the data, additional categories and subcategories emerged and therefore the list of the main categories was widened by adding:

- Outsourcing challenges with the sub-categories of requirements uncertainty and product quality;
- OSSD challenges with one sub-category, onboarding.

The data was examined to identify and classify the various challenges faced by students according to the abovementioned categories and sub-categories. It took a few iterations to reach to these final categories and sub-categories.

4 RESULT

During the project, both teams produced code that was submitted to the MOODLE HQ as candidates for future release. One team developed a Chess[9] plug-in that is now offered as a MOODLE plug-in. Furthermore, this team added new features to the Atto[10] text editor, such as creating a syntax highlighter. The other team worked on various issues, such as adding features for browsing user courses by category, creating a status indicator for online users available for messaging, adding a download progress bar, adding support for nested order on forum discussions, and many more. One of the issues was merged into the main branch; some are waiting to be peer-reviewed by the MOODLE team; whereas some are already peer-reviewed but require some additional work.

In this section, we will discuss the core GSE, the outsourcing, and the OSSD challenges students raised during the biweekly seminars as well as in their final reports. Other challenges experienced by educators while running these courses will be covered towards the end of the section.

4.1 CORE GSE Challenges

Geographic Distance. Geographic distance often creates various communication, coordination, and control issues within collaborating teams. In this course, students faced such challenges due to the distance between a student's team and the MOODLE team. Most of these issues were related to communication (including communication through documentation) with the MOODLE HQ, but students also faced some intra-team communication and coordination difficulties when they worked individually and met only through Skype, as some of the teams chose to do. These are quotes from the students:

> "First lesson that I've learned is that communication is a key factor in GSE. And also this can help avoid wrong interpretation of tasks and doing extra work."

> "Despite the many attempts to be in constant contact with the Moodle team, the communication channel between the latter and the former was rather poor."

> "The lack of transparency regarding process and communication lead to a lot of uncertainty in this project."

The team working on the MOODLE mobile app faced challenges arising from incomplete documentation:

> "Proper documentation of the process and workflow of the project as well as clear problem description when reporting issues in the tracker are problems that need to be addressed."

The other team, however, did not face any major issues in this regard because this team worked on plug-ins, which in many ways are separate elements and not part of the core focus of the MOODLE community.

> "Most features and concepts in Moodle are well documented and we were able to work on our own without any major outside help."

Some communication and coordination problems occurred between students when team members worked individually and conducted online meetings.

[9] https://moodle.org/plugins/block_chessblock.
[10] https://moodle.org/plugins/editor_atto.

> "The whole group meeting was very short, and that might be a result of participants was a bit shy. Even if we were comfortable with each other from meeting in person. We learned that a project team need some run-in time in order to have good communication in Skype."

> "Dealing with depending issues where the initial task were under-estimated became a bottleneck as others had to wait."

One student summed up communication issues as follows:

> "With more transparency, clearer issue descriptions and accessible communication channels, issues can be estimated more accurately, the team can be more productive, and features can be mainlined quicker."

Our main observation here is that students did face GSE-type communication issues during the OSSD project, and did experience the importance of continuous and timely communication when there is a geographic distance between collaborating partners. Furthermore, the students did see how essential clear and complete documentation is to avoid any misinterpretation of tasks that can lead to frustration and loss of productivity within the team.

As described above, the two GSE/OSSD teams chose different types of OSSD development work. The two teams did have similar GSE experiences even though developing a plug-in by its nature requires less coordination than adding features to an existing module, which is mostly a difference in task granularity. A plug-in is at a coarse level of granularity while adding features to an existing module is at a finer level of granularity. The biggest difference we could find was that the team working at a finer level of granularity needed to understand more of the existing MOODLE software and was suffering more from incomplete documentation.

The main lesson learned for us as GSE educators is that, if possible, one should choose OSSD projects that are highly responsive and that emphasize the importance of good documentation.

Temporal Distance. Students experienced long response times from the MOODLE HQ when submitting code or asking questions. It is difficult to say whether the long response time is due to the time difference between the local team and the MOODLE HQ, or due to the project being an OSSD project. Quotes from students:

> "In some cases, it could take several days (and in some cases no response) between any interaction from the MOODLE team after one of our team members commented on an issue. Additionally, feedback on pull requests were significantly slow where it could take over a week before any feedback was provided on pull requests."

The teams also reflected on the wider impact of the long response time from MOODLE HQ:

> "The delay is very challenging when the project is running for limited period of time, in our case one semester."

> "It is challenging to plan when you do not know when the feedback will come and how much change will it require, if any."

Considering the above mentioned challenges, our main lesson learned is to choose the open source development project carefully, picking one with an active development community.

Socio-cultural Distance. The teams experienced and reported some cultural and trust issues within each team due to the fact that the teams were multicultural, but these were rather

minor since the students had known each other for more than a year and had been working together on development projects before:

> "Even with a relatively similar group of people we experienced small cultural differences, for example, some of the international students were uncomfortable eating in front of the rest of the team while working together."

> "In the beginning the team had less work sessions which led to some degree in disorganisation and reduction in trust across the team. This has to do with the unavailability of other members when somebody needed help."

> "Our team consists of a culturally diverse team, but this did not cause any major challenge for us. Everyone quickly adopted to the diverse team and addressed any challenges as soon as they arrived."

> "Most of us had the added benefit of knowing each other before we began."

As these teams did not work closely with other open source developers involved in MOODLE project, the socio-cultural issue was not significant, which could be the case otherwise. Our main lesson learned is that an OSSD project may not expose the students to significant socio-cultural issues if the students do not work tightly with other developers in the OSS community.

Kroll et al. (2016) reported similar results from a collaborative project involving graduate students from different universities in different countries. They found that geographic, temporal, and socio-cultural distance resulted in communication, coordination, and control challenges.

4.2 Outsourcing Challenges

An OSSD project is not an outsourcing project per se. In the MOODLE governance model, the MOODLE HQ is responsible for the roadmap and for quality assurance. There are, therefore, similarities between the experiences of a local development group working jointly on MOODLE development and what an outsourced team may experience.

Requirements Uncertainty. Incomplete and ambiguous issues/specifications from the MOODLE HQ resulted in the teams doing work that was later dropped or had to be redone.

> "Communication is essential, and this became more visible in this project where the MOODLE Mobile team did, in many cases, not provide proper and clear issue descriptions."

> "Their lack of transparency and bad issue descriptions is a problem that may be found in GSD projects as well, particularly offshore outsourced projects."

> "They have poor and outdated descriptions on their issue tracker."

> "At the beginning of the project, some issues were overestimated because of creating duplications of what was already done. Some issues were interpreted incorrectly. Therefore, having someone from the Moodle team during sprint planning can improve process a lot in different aspects."

Our main observation here is that students did experience the importance of the quality of requirement specifications for software outsourcing through their participation in an OSSD project.

As mentioned earlier, the lesson learned is that one should ideally select an OSSD project that offers clear and complete documentation, to prevent requirement challenges.

Product Quality. one of the teams reported any quality management issues:

> "By strictly following the conventions and standards provided by Moodle, we believe that the software quality alignment among the team members was in place."

> "To ensure that we delivered standard and high quality solutions, it was necessary that all the team members understood the various standards defined by Moodle. Therefore, proper quality requirements of the contributions were defined."

It is more common for the main company to experience quality issues in the work product delivered by an outsourced team instead of the outsourced team itself. As some of the work products are already peer-reviewed successfully and included into the main code base, it can be considered that standards established by MOODLE HQ are clear and well defined and that both teams properly followed these standards and delivered high quality code. Our main observation, therefore, is that the students, through their participation in an OSSD project, did experience how important standards for software development are for the quality of the code produced in a software outsourcing context.

Other studies (Carmel and Agarwal 2001; Damian and Zowghi 2003) also reported similar results: that geographic distribution leads to loss of cohesion and lack of common understanding of requirements between offshore and on-site development teams. Moreover, it also impacts transfer of knowledge of requirements among personnel and developers (Helén and Nahar 2011).

Moe et al. (2012) reported the lack of quality as a main challenge and the most common reason for termination of offshore outsourcing development work in their study of three companies. Significant product quality problems were not encountered in the present study. As reported earlier, the reason may be that the standards established by MOODLE HQ are clear and well defined and that both teams properly followed these standards and delivered high quality code. This is supported by Kumar and Thangavelu (2013) that offshore and onsite teams knowledge-sharing and knowledge-transfer have significant impact on the product quality.

4.3 OSSD Challenges

Onboarding. The teams had only two formal roles: the scrum master and a build master. The build master role was important due to the MOODLE software complexity.

The teams ran stand-up meetings regularly, although not daily. Both teams ended up having stand-up meetings more regularly toward the end of the project as they saw the benefit of this when working with the quite complex software configuration of MOODLE.

Setting up the development environment was mentioned by most of the students as one of the major challenges.

> "The team often had issues when building or setting up the project."

> "Clear documentation on how to set up the project and get it up and running, is another important aspect for global software development projects that we identified while working with Moodle mobile."

> "An open source project with the aim of allowing anyone to contribute has to have proper documentation. This will allow newcomers to avoid wasting considerable amount of time (as our team did) in discovering the project setup process."

It is considered beneficial to the performance of an outsourced team to have someone from the core team working with the local team; someone who is familiar with the software architecture. The two student teams also saw this as beneficial:

> "In our case having someone from the MOODLE team—not necessary physically located with us—on early stages can accelerate the development process on early stages of the development. Someone who understands the whole architecture of the system and can explain how things work. It can prevent wasting resources and time that our team spend on experimenting and exploring things."

One of the teams faced initial problems related to the selection of issues from the MOODLE issue tracker.

> "We started with random issue picks. We have been faced with some problems because of it, for example, dependencies with other issues, issues assigned to other developers, uncertainty of the complexity, not understanding the task properly, etc. Therefore, we changed strategy by investigating more thoroughly on the issue before picking them."

Both teams encountered problems related to estimation even though students had worked on two large projects earlier and therefore had some experience in estimation for software developed locally:

> "Work estimation when dealing with open source project turned out to be problematic in our project."

> "The next challenge we faced was a high level of incorrect estimations for sprints. This was due to the team's unfamiliarity with PHP and Moodle. Only time and experience gained by working in the two had any real impact on the estimation accuracy."

> "During almost all the sprints, we had issues with problems of underestimating and overestimating issues. The best solution that we found was exploring of each task more before estimating it."

> "This could also potentially improve our issue estimates, as they were inconsistently being over and underestimated, because of lack of knowledge and poor descriptions."

Our main observation is that choosing an OSSD project with a rather large code base pushed the students to extend their skills in setting up the development environment, in picking issues for the team to work on, and in accurately estimating the efforts needed to complete these tasks. These are not GSE specific skills, but the students needed to apply their newly acquired GSE experiences when improving on these skills.

Similarly, Fagerholm et al. (2013) stated that a major challenge in an OSSD project is getting newcomers onboard into the project effectively, and concluded, based on a large collaborative program with several participating open source communities, companies, and universities, that onboarding support increases the chance for developers to be exposed to, select, and perform tasks in a proactive and self-directed manner.

4.4 Combining hands-on Experience and Research Papers

In the beginning, the teams delegated the paper reading to one or two team members who would present the paper during the seminars. This approach did not result in good discussions and deep reflection among the other students. Therefore, the structure of the paper presentation sessions was modified, requiring the group collectively to discuss the papers before the seminar and to lead the discussion during the seminar.

The discussions at the seminars and the final reports indicate that students mostly considered the paper sessions useful—even though it meant less time for development work as they had to use some of the time allocated to the course to reading papers. Some of the students found it challenging:

> "Reading scientific papers and developing at the same time leads to a bad experience as it takes our attention away from being able to focusing on development."

On the other hand, they considered the papers mostly relevant to the issues they were experiencing in their development work, but thought that some of the papers had too much overlap in terms of contributions:

> "I also found the papers we read useful, as they explained challenges and how they could be handle. Reading these side-by-side with the development made it much easier to understand the papers and the challenges the papers mentioned and the ones we as a team had to handle."

> "However, reading through papers and having experience of working on the project with the focus on global development gave me much deeper understanding of issues connected with cultural differences, trust, time zones etc."

Students generally seemed to consider the practical GSE/OSSD project to be helpful in comprehending GSE/OSSD research papers. Some students found that having to study research papers was distracting to the development work, although no student mentioned it being distracting from the experience of GSE work. Our main observation is, therefore, that a combination of practical GSE/OSSD development work and reflection on GSE/OSSD-related research papers seemed to be mutually beneficial: on one hand, the research papers gave the students the opportunity to reflect on their own experiences from the development work with a basis in the research literature. On the other hand, the research papers gave the students the possibility to reflect on GSE issues and challenges that they did not directly experience in their development project.

5 DISCUSSION INCLUDING A COMPARISON OF A GSE/OSSD WITH A TRADITIONAL GSE COURSE

In this case study, we have documented the educators' and the students' experiences from using OSSD as the basis for a GSE course that formerly was offered as a traditional GSE course with distributed teams. In this section, we will discuss and compare the two approaches to GSE education based on our experiences.

Table 2 shows the classification of the two versions of the course based on the extended GSE taxonomy, while Table 3 compares other project characteristics. As can be seen in this table, 14 students participated in the traditional GSE course. Only five of these students were working in truly global teams. The sample size is not large enough to draw firm conclusions. The study still gives us a basis for exploring the differences between the two types of GSE courses.

5.1 Students' Learning Outcome

In this section, we will compare what the students reported in the GSE/OSSD setup compared to the traditional GSE setup. The GSE/OSSD students reported that they did experience—and had to plan for how to deal with—communication, project management, requirements, and cultural challenges caused by the GSE setup. The number and types of issues, however, were somewhat different from what the traditional GSE students experienced. The differences were to a large degree caused by the differences in cross-site coordination needs and in superstructure, as discussed in Section 2.4.

Table 2. Course Classification According to the Extended GSE Taxonomy (Britto et al. 2016)

Setting	*Dimension*	*OSSD Version*	*GSE Version*
Site	Software process type	Agile	Agile
	Power distance	Small	Small
	Uncertainty avoidance	Weak	Weak
	Language distance	Medium	Medium
Relationship	Software (SW) process distance	Similar[11]	Equal
	Communication model	Low synchronicity	Balanced synchronicity
	Location	Offshore	Offshore
	Legal entity	Outsourcing	Insourcing
	Geographic distance	Far	Far
	Temporal distance	Large	Small

Table 3. Comparative Data of 2013 and 2016

	Traditional GSE	**GSE with OSSD**
Number of students	14 (5)	12
Tools	Jira, Confluence, Git, IRC	Jira, Confluence, Git, Slack
Sprint duration	4 weeks	2 weeks
Roles	Scrum Master, Product Owner, Build Master	Scrum Master, Build Master
Final product/Team Contributions	Quiz Game	Team A: a Chess plug-in, a Syntax highlighter Team B: MOODLE Mobile

Geographic Distance. Both the GSE/OSSD teams and the traditional GSE teams experienced that geographic distance was a major challenge in GSE.

For the traditional GSE team the issues were related to team members at the two locations not knowing the team members at the other location. The issues were also linked to cultural differences between the student cultures at the two locations. The students needed help from the supervisors to resolve the problems caused by these issues.

For the GSE/OSSD teams, though, the issues were mostly related to communication with the MOODLE HQ. There were two communication issues: first, the MOODLE HQ had objectives and priorities that were not clearly communicated in documents and other written information. Second, the students were experiencing long response times from the MOODLE HQ resulting in decreased team productivity. As suggested by Stroulia et al. (2011) and Fagerholm et al. (2013), some of the problems mentioned above, such as problems communicating with the MOODLE HQ, may potentially be solved by having someone from the OSSD project willing to act as a mentor and team lead for a group of students.

Temporal Distance. Time difference was not a significant issue for the traditional GSE teams because the time difference between Norway and Finland is just one hour.

The GSE/OSSD teams, however, experienced long response times. The reason is more likely due to the project being an OSSD project, and less likely due to the time difference between the MOODLE HQ and development teams. In order to avoid such issues, if possible, one should choose an open source project with an active development community, which is open to new contributors,

[11] https://docs.moodle.org/dev/Process.

and ready to engage with them on a continuous basis. Time difference could have been a more significant issue for the GSE/OSSD if there had been tighter interaction between the MOODLE HQ and the development teams.

Socio-cultural Distance. The traditional GSE teams were experiencing significant cultural challenges—mostly because of different expectations regarding workload, and project management and control.

The GSE course was part of the curriculum of an international Master's program. Some communication and coordination problems similar to those traditional GSE teams are facing were also reported within the Norwegian teams, due to the multicultural nature of the teams. One of the teams observed minor socio-cultural issues within GSE/OSSD teams although students knew each other quite well and had been working together on earlier projects. A multi-cultural team may experience cultural challenges even if they know each other and have been working together before, and should therefore be aware of potential cultural issues and be prepared to deal with such issues promptly. Moreover, student teams did not work tightly with other OSS developers, which otherwise could have potentially caused significant socio-cultural issues. The socio-cultural difference may be the cause of more issues in other OSSD projects.

Outsourcing Issues. Requirements uncertainty and product quality are often core issues in outsourcing projects. When it comes to requirements uncertainty, the traditional GSE teams were given the freedom to develop a quiz game from scratch and had one of the team members being the quiz game product owner who maintained the backlog. The GSE/OSSD teams, however, could pick tasks from the MOODLE issue tracker, and experienced problems because the requirements associated with these issues were poorly defined. It is therefore preferable to choose a project where the project's requirements are written in a clear and complete fashion in order to avoid misunderstandings and delay.

The GSE/OSSD teams reported that software quality was not a significant issue because there are well-defined conventions and standards for the produced code as well as for the development and quality assurance processes in MOODLE. The traditional GSE teams, however, had no such standards available and the teams experienced more serious product quality problems. It is crucial that projects have well-documented quality standards and guidelines for the contributing members to follow. Further, students should be encouraged to comply with these quality standards from the beginning of the project rather than this being an afterthought.

Onboarding. The GSE/OSSD teams faced the challenges of contributing to a project with a large and complex existing code base. Therefore, the process of building the software became a major challenge. The traditional GSE teams did not experience such issues. However, they did encounter challenges related to tools due to lack of experience running projects using agile methods or with the project management tools (such as Git, Confluence, and JIRA) and most of their problems were not GSE specific. It seems that the distress caused by such issues can be reduced by including a learning stage, thereby making it easier for the students to become familiar with the tools and technologies of the project. In the case of an OSSD project, having a mentor from within the project, can be an added advantage that may help the students complete this learning stage faster. Additionally, students and supervisors should be aware of the fact that OSS systems generally are larger and more complex than most of the software systems on which the students have been working, and the students are likely to experience that the team needs to spend more resources solving a task than what they initially believe.

Both types of teams also experienced estimation challenges. For the GSE with OSS teams these challenges mostly had a GSE origin, such as unfamiliarity with a large and complex code base and long response times from the MOODLE HQ. The traditional GSE teams did not have much

experience in agile projects and only some of the estimation problems were due to GSE. Students should analyze not only the complexity of the requirements when they are selecting an issue to implement, but should also be aware of its dependence on other tasks that may be unfinished and, possibly, assigned to other team/OSSD contributors.

5.2 Course Organization

During this course, we found that involving students in research paper discussions can add value to hands-on GSE/OSSD project work. Some students, however, felt that the paper session required too much of their time and attention that they would prefer to spend on development work. Unfortunately, we did not include research papers in the traditional GSE version of the course and therefore cannot elaborate on how useful research papers may be as a supplement to hands-on project work in this case.

The course organizer found it easier to organize a GSE course when the course is run within one university and not as a collaborative effort between universities. Less coordination effort was needed on course planning, teaching/supervision, and student assessment. MOODLE worked quite well as a platform for a GSE course even though a more responsive OSS community would have been beneficial to the students. The students did however acquire experience that will be useful for future participation in GSE projects.

The main challenges for a course organizer with this model are the need for onboarding support, and the lack of means to help students if they do not receive timely responses from the OSS community. In most cases, instructors would have no or little control of actions taken by the OSS community, and therefore may need to have contingency plans in place for risk management.

During the four years of running a course on GSE, we have also learned that the learning outcome depends on the student's proficiency in the agile methodology and the project management tools. Students with low proficiency have a tendency to focus more on challenges that are non-global. This is the main reason two software development courses were added to the master's program in 2015, and the reason the GSE with OSSD teams were familiar with the agile methodology and the chosen project management tools.

5.3 The Extended GSE Taxonomy

In this study, we have used the extended GSE taxonomy (Britto et al. 2016) for comparing two different GSE settings. We found the taxonomy extensions (i.e., the four site and the two relationship dimensions) relevant when comparing the two settings. We did not, however, identify any specific power distance or uncertainty avoidance challenges from the taxonomy in our settings. This could be because Norway is classified as having a small power distance index and a weak uncertainty avoidance index, as is Finland.

From an analytical point of view, we consider it an expressive weakness that the two dimensions, power distance and uncertainty avoidance, are only defined as site dimensions. In comparison, software process gives rise to two dimensions in the taxonomy, i.e., one site dimension (software process type) and one relationship dimension (software process distance). This means that the taxonomy can be used to classify the software processes at each of the sites as well as the pairwise differences in software processes at the involved sites. For power distance and uncertainty avoidance, however, there are no relationship dimensions. In our study, we could have observed larger power distance and/or uncertainty avoidance challenges, for instance, if one of the sites were classified as having large power distance and/or strong uncertainty avoidance indexes. The lack of relationship dimensions means that the taxonomy does not provide a classification of the difference in power distance (weak–weak, weak–strong, or strong–strong) or the difference in uncertainty avoidance (small–small, small–large, or large–large) between the pair of sites.

In Table 2, we have shown that a traditional GSE course may not seem too different from a GSE/OSSD course when compared according to the dimensions of the GSE taxonomy. As discussed in Section 2.4, however, such a comparison, may not address all the differences between GSE and GSE/OSSD projects that would be of importance when choosing a suitable project for a course in global software engineering. Therefore, we propose two additional relationship dimensions that should be helpful in classifying GSE (and OSSD) projects:

- **Superstructure:** This dimension specifies how individual developers and teams are connected. We propose to classify superstructures as one of:
  - **Group-group:** The project consists of two or more groups of developers—each group located at one physical site. This is the typical superstructure for GSE projects.
  - **Individual-individual:** Developers work as individuals that usually never meet physically. This is the typical superstructure for OSSD projects.
  - **Mixed:** Some developers are collocated and work in a corresponding group while some developers work remotely.
- **Cross-site coordination:** This dimension specifies the granularity of the cross-site coordination needs. We propose to classify cross-site coordination needs as one of:
  - **Fine-grained:** Project activities at the various sites are tightly coupled and need frequent coordination. This is the typical case for GSE projects.
  - **Coarse-grained:** Project activities are loosely coupled and need occasional coordination. This is the typical case for non-critical OSSD activities.
  - **Medium-grained:** Project activities are somewhat loosely coupled and need regular coordination although not so frequent. This is the typical case for critical OSSD development activities.

The superstructure for the GSE projects was *group-group* as they were typical GSE projects, whereas the superstructure chosen for GSE/OSSD projects was *mixed*, where individual MOODLE developers could contribute to the work done by our teams in collaboration with the MOODLE HQ. The cross-site coordination was *coarse-grained* in our GSE/OSSD projects due to non-critical OSSD activities. The previous years' GSE projects, on the other hand, experienced *fine-grained* cross-site coordination, as do most traditional GSE courses.

6 CONCLUSION, LIMITATIONS, AND FUTURE WORK

In this project, we have explored the integration of OSSD into a GSE course as an alternative to establishing cross-university collaboration—to what degree core GSE issues can be covered, and to what degree other global issues such as OSS itself and software outsourcing can be addressed. We have also explored the option of including research papers into the course in addition to hands-on development work. Our experience indicates that students can be exposed to challenging GSE issues in a single-university setting. The experiences students encounter, however, are different and so are the challenges that course organizers meet when compared to traditional, distributed team GSE courses. We will summarize these differences and challenges below.

Students attending a traditional, distributed team GSE course will experience core GSE challenges related to geographic, temporal, and socio-cultural distances. Our study indicates that students attending a single-university GSE course based on OSS may also experience challenges related to geographic, temporal, and socio-cultural differences. Table 2, for instance, shows that the two versions of the course had the same classification in six of the ten dimensions of the extended GSE taxonomy and only differed in four of the dimensions. Our analysis, however, shows that the extended taxonomy does not suffice in showing crucial differences between the projects. The consequences of geographic and temporal distances for the teams' GSE experiences, for instance, will

depend heavily on the suggested cross-site coordination dimension. Our GSE/OSSD project had coarse-grained cross-site coordination needs. Therefore, our GSE/OSSD students had a different experience of these distances than typical GSE students where the cross-site coordination needs are fine-grained.

GSE students will, as in other software engineering projects, also face challenges related to project management, requirements, and quality issues. Our study indicates that an OSSD project may introduce the students to other software engineering issues and challenges relevant in global settings, such as those related to software outsourcing and onboarding. In traditional distributed team GSE setups, students may experience challenges in acquiring user requirements, in choosing coding conventions and standards, and in setting up a quality assurance system. In a GSE/OSSD context, students may need to develop skills to estimate and choose tasks from an OSSD issue tracker, may be expected to learn and comply with existing coding conventions and standards, and may face a complex code base, and may need to follow detailed quality assurance procedures.

In this study, we experienced that less time and effort was spent on course organization when the course was run as a single-university, OSSD course and not a traditional, distributed team GSE course. The main challenge is the onboarding: working on open source projects can be challenging for students, especially if they have not worked in open source projects earlier or are unfamiliar with the technology. Therefore, there is a need for a learning stage before the actual implementation of tasks should start. The course organizer also needs to be aware that the students may experience challenges related to the OSSD community that the organizer cannot resolve easily because it is outside of the organizer's control. Our experience confirms the findings of Smith et al. (2014), that it is challenging for the course organizers to find and select an OSSD project of appropriate size and complexity. The MOODLE project, for instance, turned out to interact less—and less frequently—with the students than what we expected.

In this study, we found that students had different opinions on the benefits of including research papers into a GSE course. Some students would rather spend their time and efforts on the development work only, while other students saw the combination of studying research papers and doing development work as mutually beneficial. Most student feedback, however, was positive and suggested that combining actual GSE project work with studying research papers added an additional value to the experience. This suggests that GSE course designers should consider adding research papers to cover relevant issues that students may not experience directly in their hands-on projects, such as onboarding, outsourcing, and OSSD, to help students understand a larger part of the global software engineering picture.

One limitation of this study is the results are based on the data collected from only one site. It was not feasible to seek out the experiences of developers at MOODLE HQ. As they are the ones who are ultimately responsible for product quality and maintenance, it would be valuable to gather their view of the challenges and to compare that with the present results. Case studies are often criticized for lack of reliability, due to researcher subjectivity, for instance, and generalizability. Case studies are still considered a methodology that can be used to advance research where there are no other easily applicable methodologies. We acknowledge that studying just one course with a limited number of students is a rather weak basis for strong conclusions and recommendations. However, we still think our case study suggests that using OSSD as a basis for advanced software engineering education can successfully give the students experience in dealing with core global collaboration issues. At the same time, OSSD projects may expose the students to issues that are relevant in the global software development world but are usually not faced in traditional GSE education.

As mentioned at the very beginning of this article, OSSD has been suggested as the basis for GSE education (Beecham et al. 2017). In this case study, we have observed that students did get to experience some core GSE challenges as well as other challenges related to global software

engineering. Our main lesson learned, however, is that the nature of the OSS project and the community will influence, and may limit, what GSE challenges the students will meet. We also learned that the teaching staff has little control over what happens inside the community. In the future, we would want to see more research on the combination of OSSD and GSE—especially on projects run together with OSS communities that are active and responsive to the students' requests and contributions to make the students experience more of the issues related to fine-grained cross-site coordination.

ACKNOWLEDGEMENT

Our appreciation goes to the students of this course whose insights and comments helped in this study. We also like to thank the editor, associate editors, and reviewers for their great efforts in improving the quality of this article.

REFERENCES

Jodi Aronson. 1994. A pragmatic view of thematic analysis. *The Qualitative Report* 2, 1 (1994), 1–3.

Jeffry Babb, Rashina Hoda, and Jacob Nørbjerg. 2014. Embedding reflection and learning into agile software development. *IEEE Software* 31, 4 (2014), 51–57.

Sebastian Barney, Claes Wohlin, Panagiota Chatzipetrou, and Lefteris Angelis. 2011. Offshore in-sourcing: A case study on software quality alignment. In *Proceedings of the 2011 IEEE 6th International Conference on Global Software Engineering (ICGSE'11)*. IEEE Computer Society, Washington, DC, 146–155.

Sarah Beecham, Tony Clear, Daniela Damian, John Barr, John Noll, and Walt Scacchi. 2017. How best to teach global software engineering? Educators are divided. *IEEE Software* 34, 1 (2017), 16–19.

Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative Research in Psychology* 3, 2 (2006), 77–101.

Ricardo Britto, Claes Wohlin, and Emilia Mendes. 2016. An extended global software engineering taxonomy. *Journal of Software Engineering Research and Development* 4, 1, Article 3 (2016).

Erran Carmel and Ritu Agarwal. 2001. Tactical approaches for alleviating distance in global software development. *IEEE Software* 18, 2 (2001), 22–29.

Valentine Casey. 2010. Virtual software team project management. *Journal of the Brazilian Computer Society, Special Issue on Global Software Development* 16, 2, 83–96.

Tony Clear, Sarah Beecham, John Barr, Mats Daniels, Roger McDermott, Michael Oudshoorn, Airina Savickaite, and John Noll. 2015. Challenges and recommendations for the design and conduct of global software engineering courses: a systematic review. In *Proceedings of the 2015 ITiCSE on Working Group Reports (ITICSE-WGR'15)*. 1–39.

Daniela Damian and Didar Zowghi. 2003. RE challenges in multi-site software development organisations. *Requirements Engineering* 8, 3 (2003), 149–160.

Fabian Fagerholm, Patrik Johnson, Alejandro Sánchez Guinea, Jay Borenstein, and Jürgen Münch. 2013. Onboarding in open source software projects: A preliminary analysis. In *Proceedings of the 2013 IEEE 8th International Conference on Global Software Engineering Workshops*. 5–10.

Kevin Gary, Andinet Enquobahrie, Luis Ibanez, Patrick Cheng, Ziv Yaniv, Kevin Cleary, Shylaja Kokoori, Benjamin Muffih, and John Heidenreich. 2011. Agile methods for open source safety-critical software. *Software: Practice and Experience* 41, 9 (2011), 945–962.

Olly Gotel, Vidya Kulkarni, Long Chrea Neak, Christelle Scharff, and Sopheap Seng. 2007. Introducing global supply chains into software engineering education. In *Proceedings of the 1st International Conference on Software Engineering Approaches for Offshore and Outsourced Development (SEAFOOD'07)*, Bertrand Meyer and Mathai Joseph (eds.). Springer-Verlag, Berlin, 44–58.

Greg Guest, Kathleen M. MacQueen, and Emily E. Namey. 2012. *Applied Thematic Analysis*, 1st ed. SAGE Publications.

Milla Helén and Nazmun Nahar. 2011. Key barriers of globally distributed software products development. In *Proceedings of the 2011 PICMET Conference: Technology Management in the Energy Smart World (PICMET'11)*. 1–13.

Reid Holmes, Michelle Craig, Karen Reid, and Eleni Stroulia. 2014. Lessons learned managing distributed software engineering courses. In *Companion Proceedings of the 36th International Conference on Software Engineering (ICSE Companion '14)*. ACM, New York, 321–324.

Philipp Holtkamp, Jussi P. P. Jokinen, and Jan M. Pawlowski. 2015. Soft competency requirements in requirements engineering, software design, implementation, and testing. *Journal of Systems and Software* 101 (2015), 136–146.

Philipp Holtkamp and Jan M. Pawlowski. 2015. A competence-based view on the global software development process. *Journal of Universal Computer Science* 21, 11 (2015), 1385–1404.

William L. Honig and Tejasvini Prasad. 2007. A classroom outsourcing experience for software engineering learning. *SIGCSE Bulletins* 39, 3 (2007), 181–185.

Emam Hossain, Muhammad A. Babar, and June Verner. 2009. How can agile practices minimize global software development co-ordination challenges? In *Proceedings of the European Conference on Software Process Improvement (EuroSpi'09)*. Springer, 81–92.

Ronald Jabangwe and Darja Šmite. 2012. An exploratory study of software evolution and quality: before, during and after a transfer. In *Proceedings of the 2012 IEEE 7th International Conference on Global Software Engineering*. 41–50.

G. Kannabiran and K. Sankaran. 2011. Determinants of software quality in offshore development—an empirical study of an Indian vendor. *Information and Software Technology* 53, 11 (2011), 1199–1208.

Josiane Kroll, Ita Richardson, and Jorge L. N. Audy. 2014a. FTS-SPM: a software process model for follow the sun development: preliminary results. In *Proceedings of the 2014 IEEE International Conference on Global Software Engineering Workshops*. 21–26.

Josiane Kroll, Ita Richardson, and Jorge L. N. Audy. 2014b. Proposing a software process model for follow the sun development. In *Proceedings of the 26th International Conference on Software Engineering & Knowledge (SEKE'14)*. 412–415.

Josiane Kroll, Caroline Q. Santos, Leticia S. Machado, Sabrina Marczak, and Rafael Prikladnicki. 2016. Challenges and lessons learned on preparing graduate students for GSE work: Brazilians' perceptions on a multi-site course experience. In *Proceedings of the 11th IEEE International Conference on Global Software Engineering Workshops (ICGSE Workshops'16)*. 37–42.

S. Arun Kumar and Arun K. Thangavelu. 2013. Factors affecting the outcome of global software development projects: an empirical study. In *Proceedings of the 2013 International Conference on Computer Communication and Informatics*. 1–10.

Mary C. Lacity, Shaji A. Khan, and Leslie P. Willcocks. 2009. A review of the IT outsourcing literature: Insights for practice. *Journal of Strategic Information Systems* 18, 3 (2009), 130–146,

Luigi Lavazza, Sandro Morasca, Davide Taibi, and Davide Tosi. 2010. Applying SCRUM in an OSS development process: an empirical evaluation. In *Agile Processes in Software Engineering and Extreme Programming (XP'10)*, A. Sillitti, A. Martin, X. Wang, E. Whitworth (Eds.). Springer, Heidelberg, 147–159.

James Long. 2010. Outsourcing in next generation technical software engineering education. In *Proceedings of the 2010 American Society for Engineering Education Annual Conference and Exposition*. 3364–3376.

Alok Mishra and Ali Yazici. 2011. An assessment of the software engineering curriculum in Turkish universities: IEEE/ACM guidelines perspective. *Croatian Journal of Education* 13, 1 (2011), 188–219.

Deepti Mishra and Alok Mishra. 2012. A global software inspection process for distributed software development. *Journal of Universal Computer Science* 18, 19 (2012), 2731–2746.

Deepti Mishra and Alok Mishra. 2011. A review of non-technical issues in global software development. *International Journal of Computer Applications in Technology* 40, 3 (2011), 216–224.

Deepti Mishra and Alok Mishra. 2010. A software inspection process for globally distributed teams. In *On the Move to Meaningful Internet Systems: OTM 2010 Workshops (OTM'10)*, R. Meersman, T. Dillon, P. Herrero (Eds.), Lecture Notes in Computer Science, Vol. 6428. Springer, 289–296.

Nils B. Moe, Darja Šmite, and Geir K. Hanssen. 2012. From offshore outsourcing to offshore insourcing: Three stories. In *Proceedings of the 2012 IEEE 7th International Conference on Global Software Engineering*. 1–10.

Paul N. Nkwi, Isaac K. Nyamongo, and Gary W. Ryan. 2001. *Field Research Into Socio-cultural Issues: Methodological Guidelines*. Research International Center for Applied Social Sciences, Research, and Training.

John S. Persson, Lars Mathiassen, Jesper Boeg, Thomas S. Madsen, and Flemming Steinson. 2009. Managing risks in distributed software projects: an integrative framework. *IEEE Transactions on Engineering Management* 56, 508–532.

Justus J. Randolph. 2007. *Multidisciplinary Methods in Educational Technology Research and Development*. Hämeenlinna, Finland, Häme University of Applied Science Press.

Ita Richardson, Valentine Casey, Fergal McCaffery, John Burton, and Sarah Beecham. 2012. A process framework for global software engineering teams. *Information and Software Technology* 54, 11 (2012), 1175–1191.

Bjarne R. Schlichter and John S. Persson. 2014. Trust in co-sourced software development. In *Proceedings of the Mediterranean Conference on Information Systems, Verona, Italy*. 1–11.

Suprika V. Shrivastava and Urvashi Rathod. 2015. Categorization of risk factors for distributed agile projects. *Information and Software Technology* 58 (2015), 373–387.

Outi Sievi-Korte, Kari Systä, and Rune Hjelsvold. 2015. Global vs. local—experiences from a distributed software project course using agile methodologies. In *Proceedings of the 2015 IEEE Frontiers in Education Conference*.

Darja Šmite. 2014. Distributed project management: Ten misconceptions that might kill your distributed project. In *Software Project Management in a Changing World*, G. Ruhe and C. Wohlin (Eds.). Springer Verlag, 301–320.

Darja Šmite, Claes Wohlin, Zane Galviņa, and Rafael Prikladnicki. 2014. An empirically based terminology and taxonomy for global software engineering. *Empirical Software Engineering* 19, 1 (2014), 105–153.

Thérèse Mary Smith, Robert McCartney, Swapna S. Gokhale, and Lisa C. Kaczmarczyk. 2014. Selecting open source software projects to teach software engineering. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE'14)*. 397–402.

Anne-Marie Søderberg, S. Krishna, and Pernille Bjørn. 2013. Global software development: commitment, trust and cultural sensitivity in strategic partnerships. *Journal of International Management* 19, 4 (2013), 347–361.

Tine Sonne, Anne-Marie Søderberg, and Thomas Tøth. 2013. Negotiating and spanning boundaries in offshore outsourcing: Indian and Danish perceptions of intercultural collaboration. In *Proceedings of the International Academy of International Business Annual Meeting 2013 (AIB'13)*.

Eleni Stroulia, Ken Bauer, Michelle Craig, Karen Reid, and Greg Wilson. 2011. Teaching distributed software engineering with UCOSP: the undergraduate capstone open-source project. In *Proceedings of the 2011 Community Building Workshop on Collaborative Teaching of Globally Distributed Software Development (CTGDSD'11)*. ACM, New York, 20–25.

Ilkka Tervonen, Antti Haapalahti, Lasse Harjumaa, and Jouni Similä. 2013. Outsourcing software testing: A case study in the Oulu area. In *Proceedings of the 2013 13th International Conference on Quality Software*. 65–74.

Received April 2017; revised April 2018; accepted May 2018