This is a post-peer-review, pre-copyedit version of this article. The final authenticated version is available online at:
https://doi.org/10.1109/ICSE.2015.157

Learning Global Agile Software Engineering Using Same-Site and Cross-Site Teams

Maria Paasivaara*, Kelly Blincoe†, Casper Lassenius*, Daniela Damian†
Jyoti Sheoran†, Francis Harrison†, Prashant Chhabra†, Aminah Yussuf†, Veikko Isotalo*

*Software Process Research Group, Aalto University
FI-00076 Aalto, FINLAND
firstname.lastname@aalto.fi

†SEGAL, University of Victoria
Victoria, BC, Canada
kblincoe@acm.org, danielad@cs.uvic.ca

***Abstract*—We describe an experience in teaching global software engineering (GSE) using distributed Scrum augmented with industrial best practices. Our unique instructional technique had students work in both same-site and cross-site teams to contrast the two modes of working. The course was a collaboration between Aalto University, Finland and University of Victoria, Canada. Fifteen Canadian and eight Finnish students worked on a single large project, divided into four teams, working on interdependent user stories as negotiated with the industrial product owner located in Finland. Half way through the course, we changed the teams so each student worked in both a local and a distributed team. We studied student learning using a mixed-method approach including 14 post-course interviews, pre-course and Sprint questionnaires, observations, meeting recordings, and repository data from git and Flowdock, the primary communication tool. Our results show no significant differences between working in distributed vs. non-distributed teams, suggesting that Scrum helps alleviate many GSE problems. Our post-course interviews and survey data allows us to explain this effect; we found that students over time learned to better self-select tasks with less inter-team dependencies, to communicate more, and to work better in teams.**

I. Introduction

Challenges arise when software development is distributed across multiple geographic locations, referred to as Global Software Engineering (GSE). Communication is particularly difficult in GSE due to the time-zone differences, geographic distances and cultural differences. Despite their emphasis on face-to-face communication, agile development methods have been increasingly adopted in distributed settings in industry. Agile methods emphasize frequent communication, transparency of progress and short iterations, and they have been found to address many challenges faced in GSE [1]–[3].

In line with this shift in industry trends, examples of software engineering curricula teaching GSE using agile methods have emerged [4]–[7]. The reported courses are practical, project-driven, utilize agile development methods like Scrum [8] and are arranged in collaboration between two or more universities, enabling students to learn GSE in realistic settings [9]. In our project-driven GSE course, we found that the use of agile methods, augmented with GSE best practices, helped the students to learn important GSE competencies [7]. Scrum has short Sprints with several formal meetings for Sprint Planning, Demos and Retrospectives, as well as emphasizes early and frequent communication within and between teams, and with the Product Owner (PO). Using such a process allows the students to experience many phases of the life-cycle of a distributed project, while reflecting on and continually improving their team working practices. In our previous edition of the course [7], students worked only in distributed teams. In the latest instance of the course, described in this paper, they experience working both in distributed, as well as local Scrum teams, since both modes of working are common in industry.

In this paper, we describe the latest edition of our GSE course, taught in collaboration between Aalto University, Finland and University of Victoria, Canada. Similar to the past edition of this course, students worked on a single large project for an industrial customer utilizing distributed Scrum and GSE best practices, such as site visits, frequent synchronization, and multiple communication modes. New to this years edition of the course, we structured the project so that each student spent, after the training Sprints, two two-week Sprints working in a same-site team and two two-week Sprints working in a cross-site team to better understand the differences between the two modes of working. Using post-course interviews, iteration questionnaires and repository data, we describe the differences students reported in working in same-site and cross-site teams, examine how the students' GSE competencies improve over time, and how the students perceived the new course format.

We found that students learned important GSE competencies throughout the course including the ability to select tasks that minimized inter-team dependencies, reducing coordination overhead. Surprisingly, students did not perceive major differences between working in local and distributed teams and, in some cases, even preferred the distributed teams, since all communication was easily traceable within Flowdock[1], the teams' preferred mechanism for communication. Flowdock

[1]http://www.flowdock.com

was used also in the local teams, but the distributed teams conformed better to the agreed communication practices, as students recognized their importance in particular when working in a distributed manner.

II. Previous Work

A. GSE Challenges and the Agile Solution

GSE brings many challenges including cultural, language and background differences, lack of overlapping working hours, lack of frequent contact, lack of trust and lack of willingness to communicate openly [10]. Communication is impacted by each of these challenges, making it the single biggest challenge of GSE.

Agile development methods, and Scrum in particular, are becoming mainstream in industrial GSE [1], [2]. Agile methods have been found to address challenges deriving from a distributed way of working [2]. Scrum emphasizes communication and provides a structure for frequent formal communication both within and between the teams, as well as with the Product Owner through Daily Scrums, Scrum-of-Scrums, Sprint Plannings, Demos and Retrospectives. Daily Scrums are brief daily status meetings for each Scrum team. When several teams are working on the same project, Scrum-of-Scrums meetings are held for inter-team coordination. Teams meet in the beginning of each Sprint in Sprint Planning meetings to plan and estimate the new user stories to be implemented. Demos and Retrospectives are held at the end of each iteration to showcase newly implemented functionality and to improve the process [8]. Research on the usage of Scrum in distributed projects has found these formal meetings encourage team members between sites to communicate frequently and facilitate informal communication after the meetings, as participants notice that there are issues needed to be solved or questions to ask [3]. A systematic literature review found that modified Scrum practices can help mitigate communication challenges resulting from time-zone differences [1] .

B. Teaching GSE

The GSE teaching literature has identified skills that GSE professionals need and are, thus, important to be taught to students learning GSE (e.g. [11], [12]): regular communication with distributed team members [12], [13], team dynamics [12] working in culturally diverged teams [13], managing time [13], and using collaborative technologies [13]. Notably, all of these are soft skills. Product architecture is another important aspect of GSE since, arguably, modular architectures offer the potential to reduce cross-site communication [14]. However, the evolving nature of designs over the course of a project limits a design's ability to predict and optimize project cross-site communication [15]. The need to respond to changing architectural dependencies further emphasizes the importance of developing communication skills in GSE courses.

Teaching the soft skills required for GSE is best done through practical experience in which students can learn by doing [11]. A systematic mapping study of 19 GSE courses [9] found only one course that was theoretical in nature, while all others had a distributed project as a central element. Despite the fact that agile methods are gaining popularity in industry, most reported GSE courses still use a waterfall process. Moreover, only a subset of the reported courses teach students the whole life-cycle of a software development project. In fact, the systematic literature review [11] concludes that it is not possible for instructors to cover all of the stages of GSE and therefore one specific aspect should be focused on. As evident from this and our previous papers, we do not share this view.

In the previous instance of our course, we found that the Scrum method, along with supporting collaboration practices and tools, supports the learning of important GSE competencies, such as distributed communication and teamwork, building and maintaining trust, using appropriate collaboration tools, and inter-cultural collaboration [7]. Scharff et al. [4]–[6] found that Scrum increased the transparency of both the process and the developed software product in their GSE course using Scum practices over three years after transitioning from teaching the course using the waterfall methodology.

We are not aware of any reports of courses where the students are members of both local and distributed teams. In this paper, we describe our experience in structuring the course project in this way and add to the empirical evidence that students learn strategies to overcome GSE challenges through self-selection in project task allocations.

C. Sociotechnical Congruence

In large and distributed software projects, developers often work on tasks in parallel. When technical dependencies exist between these tasks, developers often need to coordinate their work. When work dependencies exist and developers do not coordinate, productivity and quality problems can occur, including increased task resolution time, increased software faults, build failures, redundant work, and schedule slips [16]–[19]. Therefore, when coordination is focused between the team members with work dependencies (coordination needs), the project can see performance and quality benefits. Cataldo et al. [16] conceptualised a way to measure the extent to which coordination needs and coordination behavior are aligned in practice, called Socio-Technical Congruence (STC). STC is expressed as the ratio between coordination needs that are satisfied by actual acts of coordination and the set of all coordination needs. For example, if a team has technical dependencies resulting in 10 coordination needs between the developers and 4 pairs of those developers coordinate their work, the STC score would be 4/10 or 0.4. In this paper, we use this STC measure to analyze how coordination between the students aligns with their coordination needs.

III. Course Design

A. Learning Goals

Our main goal, similar to our previous course, was to teach students GSE competences by emulating a modern real-world GSE environment through the use of agile practices that involve significant inter-team, inter-site and inter-cultural communication, as well as to teach and expose students to

various collaborative technologies that provide infrastructural and cognitive support for effective communication in global teams. In addition, we wanted students to experience working both in site-specific (local) as well as cross-site (distributed) Scrum teams. Reflecting upon the differences between the two teams physical structures, through surveys and process introspection, was one of our additional instructional strategies in this edition of the course.

B. Course Setup

We ran the course jointly between Aalto University, Finland, and University of Victoria, Canada, over a 3-month period, from January to April 2014, with slightly different configurations at the two locations. The course at Aalto University was a capstone project-based course without lectures, while the University of Victoria course setup included a more traditional structure in which students participated in lectures, as well as in-class and online discussions of experience. The Finnish course started in September 2013, and the Canadians joined in January 2014.

In total, we had 23 students participating, 15 in Canada and 8 in Finland. The students worked on a single large project, building a mobile client from scratch for Agilefant[2], an existing backlog management system developed at Aalto University, and now commercialized by a university spin-off.

C. Project Process: Distributed Scrum

Similar to our previous course, we used Scrum [8], [20] as the basic process framework. We organized the students into four scrum teams, each consisting of 4–6 members. Each team had a Scrum Master (SM), which changed every Sprint. The teams self-selected their SMs, e.g. based on team member's skills and preferences. The Product Owner (PO) for all teams was a representative from the spin-off company commercializing Agilefant, located in Finland. Thus, the PO can be seen as external industrial customer.

The first two Sprints, which are not analyzed in our case study, were each one week long. Their purpose was to familiarize the students with the project codebase, the Scrum methodology, and the development and communication tools. Sprint 0 involved only Canadian teams. Sprint 1 had four global teams, each distributed between Finland and Canada.

As illustrated in Figure 1, the remaining four Sprints were each two-weeks long and each had two global software teams (a mix of Canadian and Finnish students) and two local software teams where all members were from the same site. To accomplish this, two of the globally distributed teams from Sprint 1 remained intact for Sprints 2 and 3. The remaining two teams were split into two local teams with all Canadian students on one team and all Finnish students on the other team. In Sprints 4 and 5, teams were switched. The members of the two global teams from the previous Sprints were converted into two local teams and the students from the two local teams reverted back to their Sprint 1 global teams.

[2]http://www.agilefant.com

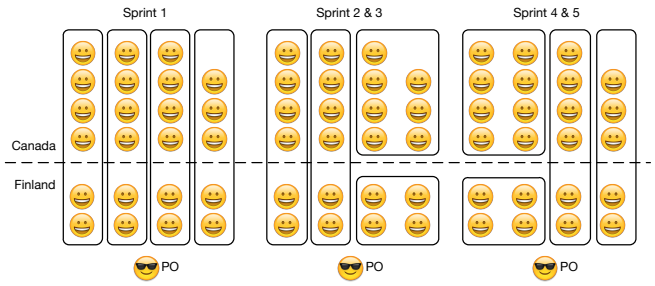


Fig. 1: Team Setup. There were four distributed teams in Sprint 1. In Sprints 2&3, two teams remained in tact, while the other students organized into local teams. The setup was reversed for Sprints 4&5.

Each Sprint started with a synchronous Sprint Planning meeting for all teams organized via Google Hangout, held in each locations' open work area (termed war rooms in the agile methodology). The PO presented an overview of the user stories to be developed in the Sprint, then the teams self-selected their user stories. After story selection, the teams split into separate videoconference sessions to estimate the user stories and perform an initial task breakdown. The PO connected with each distributed team individually and explained the user stories in more detail, giving the team a basis for story-point estimation. Immediately after these team-specific sessions, each team communicated their plan to the PO and each other in another joint videoconference session.

During the Sprints, teams held two weekly 15 minute videoconference Daily Scrum meetings, followed by informal hangout sessions. The needs and topics for these informal follow-up discussions arose from Daily Scrum meetings. Since the time allocated to this project was 10–12hours/week/student, this frequency for Daily Scrums was determined as adequate. In addition to these required meetings, students were encouraged to have informal meetings whenever they felt it was necessary.

At the end of each Sprint, each team held a Retrospective meeting, using Google Hangout when the team was distributed, to reflect and improve their ways of working, followed by a joint Demo to the PO and the other teams in the two connected war rooms. The Demos, Retros and Sprint Planning meetings all took place every second week during a time-boxed 2.5 hour slot. The order of the team specific Sprint Planning sessions and Retros varied from team to team, so that the teams could use the PO's time efficiently.

According to the GSE best practices, we encouraged the students to use multiple communication modes and suitable tools to support different communication and collaboration purposes. The main tools used were Google Hangout, GitHub, git, AgileFant, and Flowdock. Moreover, in accordance with GSE best practices [21], the Finnish instructor, the PO and two Finnish students visited Canada for ten days in January 2014, when the Canadian course commenced. During the visit, Scrum training, as well as technical training was given. During this time, Sprint 0 was run, and Sprint 1 started.

IV. Methodology

The aim of the research presented in this paper was to study the new course format for teaching students GSE by having them work in both local and globally distributed Scrum teams in order to learn the differences between the two modes of work. In addition, we were interested in studying student learning over time as the course progressed.

We posed the following research questions:

- RQ1 Did students experience differences in working in global vs. local teams, and if so, what?
- RQ2 Did students learn effective strategies for dealing with GSE challenges over time?
- RQ3 How did the students perceive the new course format?

For each of these research questions, we considered the following GSE elements: Scrum practices, communication, teamwork, inter-team collaboration, and task allocation. For this analysis, we collected data using a mixed-method approach combining qualitative and quantitative methods [22]. We used surveys, observations, video recordings of meetings, collection of communication data, and interviews to triangulate results. In addition, we collected data from tools used during the course, including GitHub, git, AgileFant, and Flowdock. The collected data is shown in Table I.

A. Asynchronous Communication Network

To identify which users communicated with each other both within and across teams, we analyzed the Flowdock communication instances. Flowdock was the most commonly used tool for all course communication.

Flowdock allows users to tag other users within messages to indicate whom the message is directed to. This feature was used consistently in Sprints 3-5 allowing us to automatically determine communication links between users for these Sprints. As tags were not utilized consistently in Sprint 2, we manually inspected each message to identify which users each message was directed to. We also manually inspected the approximately 15% of messages from Sprints 3-5 which did not contain any tags. Once the message recipients were established, we created directed communication links between each message's sender and the message recipients.

After the course, two students with intimate knowledge of the user stories manually mapped each message to its associated user story or stories by considering the message content and the description of each user story.

B. Sociotechnical Congruence

STC is the ratio of satisfied coordination needs to all coordination needs. We computed coordination needs between users by considering the technical dependencies between user stories. In cases where users worked on the same user story or on dependent user stories, we interpreted it as a coordination need. Users were linked to user stories if they were listed as the assignee within Agilefant or if they were the author of one of the commits associated with the user story.

To identify dependencies between user stories, we considered the technical dependencies occurring between artifacts involved in those user stories through logical coupling [23]. Logical coupling tracks files that have been historically checked in together and aims at identifying semantic relationships that may not manifest in the syntax of the programmatic implementation of the software product. For this case study, files were considered dependent if they had historically been checked in together four or more times. Artifacts were associated to user stories by considering all development artifacts (JavaScript, HTML, CSS and XML files) involved in commits related to a user story. The mappings were done using a combination of automated and manual work.

C. Surveys

Prior to the project start, students filled out a background questionnaire with questions on their prior experiences with software projects and agile methods, as well as their expectations for the course. At the end of each Sprint, students filled out a questionnaire about their view of the Scrum process and practices, project and team success, trust [24], teamwork quality [25], [26], main challenges experienced, and main learnings during the last Sprint.

D. Post-course interviews

After the course, 12 students, 4 from Finland and 8 from Canada, were interviewed about their experiences and learning. In addition, we interviewed the Finnish PO. The interviews were semi-structured and lasted for about one hour. The discussions were recorded, transcribed and analysed by thematic coding in the qualitative analysis software Atlas.ti[3].

V. Results

In this section, we describe our survey, interview and repository analysis results on the following GSE elements: Scrum practices, communication, teamwork, inter-team collaboration, and task allocation. We also present the student reaction to this new course format.

A. Scrum Practices

As explained in the course design, the teams were required to use all Scrum practices and roles during the project.

In the iteration surveys, we asked students to evaluate the Scrum process itself and various Scrum rituals. Our findings, shown in Figure 2a, indicate that students were satisfied with Scrum and the Scrum practices, something we expected based on our previous experiences [7], and also corroborated in the interviews, as the following quote shows.

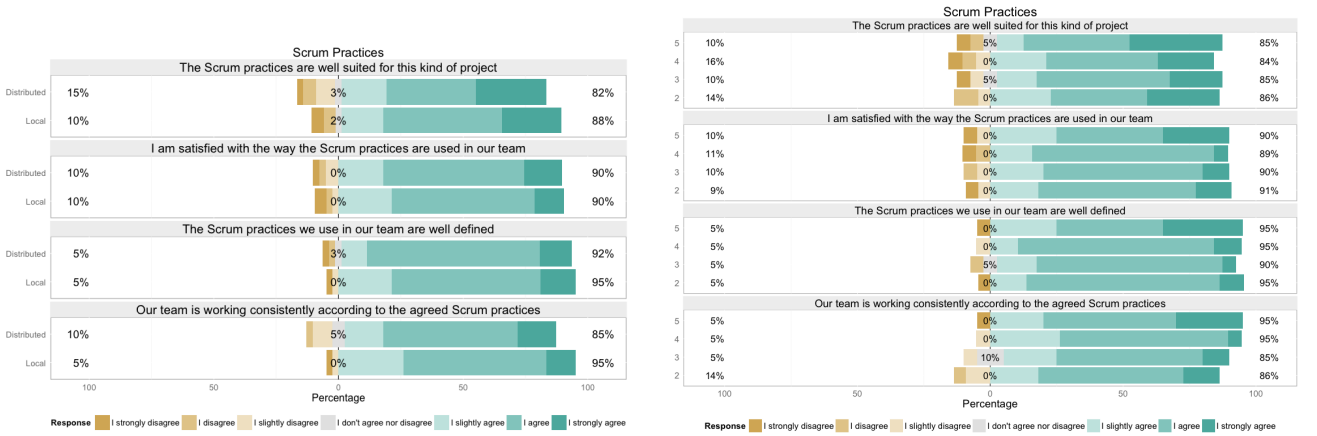
> *I think it's [Scrum] good, especially for this project when we have some distributed teams. [...] It keeps everybody working. You can't slack. And it keeps the project going in the [sic] good direction. It makes everybody to become more and more engaged in this project.* — Team Member, Canada

Quite surprisingly, however, the differences between satisfaction with the Scrum practices were not significant when working in local vs. distributed teams, see Figure 2a. While

[3]http://www.atlasti.com

TABLE I: Data Collection

Data collection	Purpose / Instruments	Data collected
Pre-course survey	Student background, expectations	23 responses
Sprint surveys	Scrum practices, Trust, Teamwork, Communication	20-23 responses
Flowdock communication	Communication	Actors, Messages
Observations	Teamwork, communication	5 Demos & Sprint Planning meetings, Retros, Daily Scrums, Teamwork in war rooms
Student logs	Communication, encountered issues, tasks, learning	
Agilefant	Estimation accuracy, work breakdown	Task estimates and actuals, burndown
Interviews	Learning, communication, communication tools, improvement ideas	13 interviews, 40-90 min each (8 Canadian & 4 Finnish students, Finnish PO)



(a) Distributed vs. Local Teams

(b) Scrum Practices over Time (Labels on the left indicate Sprint numbers 2–5)

Fig. 2: Students Experiences with Scrum

not statistically significant, nor surprising, students reported that the Scrum practices were slightly less suited to working in a distributed team than in a local one and that the process conformance was slightly poorer when the teams were distributed. However, in the interviews they stated the opposite: several students mentioned that in distributed teams they were actually following the agreed practices a bit better than in the collocated teams, e.g., communicating right away in Flowdock when working on tasks or having problems, instead of waiting for the next face-to-face meeting, which did happen when working in a local team.

When interpreting this data, it is good to keep in mind that the course required all Scrum practices to be used, giving the teams fairly little leeway in adapting the practices.

Looking at the Scrum practices over time, Figure 2b, the situation looks quite stable, with high levels of satisfaction throughout the course, and slightly better process conformance over time, which seems quite natural, as students gained more experience with the process and practices.

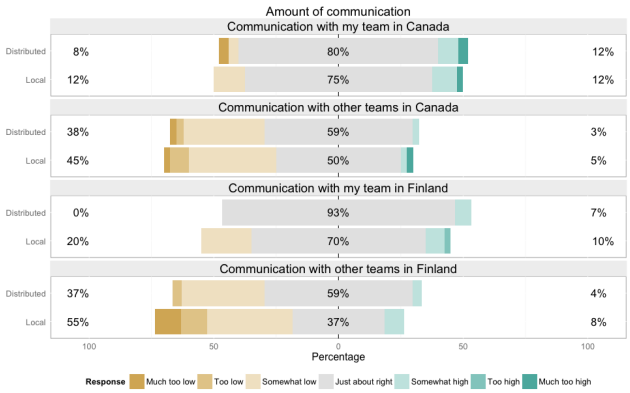
B. Communication

The Scrum process forces people to communicate by having many meetings. Thus, a big part of the communication both inside and between our Scrum teams took place in these formal synchronous meetings: local meetings were normally face-to-face, whereas global meetings were arranged using videoconferencing with Google Hangout. In addition to Google Hangout, the teams used several tools for informal communication: email, Flowdock, Agilefant, GitHub, Git and even phones.

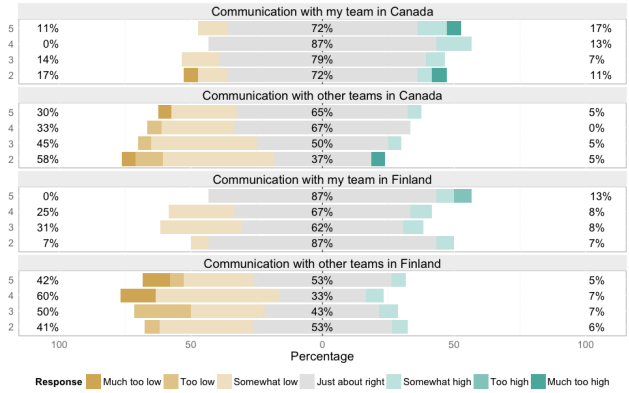
According to our interviews, Flowdock emerged as the most important communication forum after the synchronous video meetings. Flowdock was used as a chat tool to let other team members, distant or local, know what each team member was working on and to ask questions. In Flowdock, all communication is visible to everyone, but you can also target messages to specific recipients using tags. While these messages are still visible to everyone, the person(s) tagged in the message receives a notification through email or the mobile app on their phone ensuring they are aware of the message.

We examined the communication that occurred within Flowdock to identify how the teams fulfilled their coordination needs. We saw that over time, the students fulfilled a greater number of their coordination needs through communication as evidenced by the increasing STC scores shown in Table II. As shown in the table, the students did not increase their communication, rather their coordination needs were reduced and their coordination became more focused.

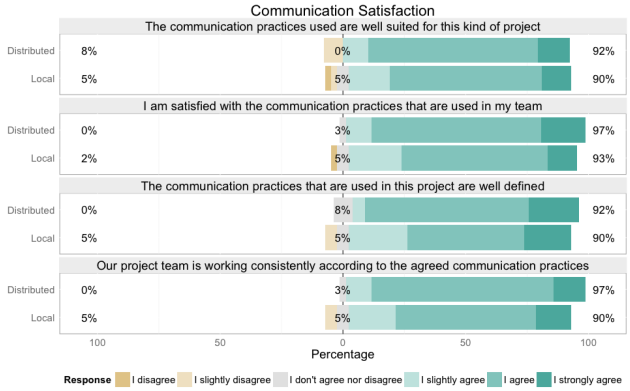
According to the iteration surveys, the students were very satisfied with the amount of communication inside their project team (Figures 3a and 3b). Moreover, from Figures 3c and 3d we can see that the students were highly satisfied with com-



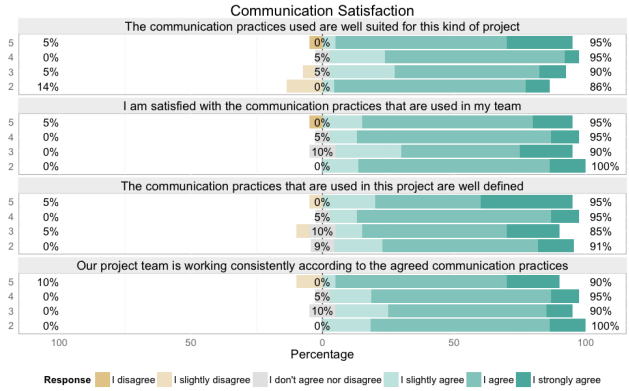
(a) Distributed vs. Local



(b) Subjective Communication Amount over Time (Labels on the left indicate Sprint numbers 2–5)



(c) Communication Satisfaction in Distributed vs. Local Teams



(d) Communication Satisfaction over Time (Labels on the left indicate Sprint numbers 2–5)

Fig. 3: Communication Satisfaction and Subjective Amount

munication practices used in the project and in their own team. We saw almost no differences between local and distributed teams. The only slight difference was that the students reported that their team better followed the agreed communication practices when distributed. This finding was confirmed by the interviews and the reason given was that the students thought it more important to follow the agreed practices when distributed, while local teams could more easily have ad-hoc communication. In the second iteration with the same team (Sprints 3 and 5), the students felt they were following slightly better the agreed communication practices. The high satisfaction with communication practices and having almost no difference between local and global teams seems to indicate that the Scrum communication practices are well suited for distributed teams and that Scrum practices might mitigate the communication differences between local and global teams.

C. Teamwork

In the post-course interviews, the students reported that the course project highlighted the big difference between working in a distributed and a local team: in a local team, you could receive help and answers fast, and face-to-face interactions encouraged trust.

> *It became so much simpler when we had [a local team in Finland] because we could see each other so often and we did not need to think, whether the Canadians were sleeping or something like that...* — Team Member, Finland

We were positively surprised that the students did not report big challenges when working in distributed teams. According to the interviews, there seems to be two clear reasons for that: 1) the Scrum rituals, such as Sprint planning meetings, daily Scrums and Scrum-of-Scrum meetings, supported by good communication and coordination tools, such as Flowdock and Agilefant, helped the team members to stay informed, and 2)

TABLE II: Sociotechnical Congruence over Time

Sprint	Fulfilled Coordination Needs	Coordination Needs	STC
2	6	65	0.09
3	44	220	0.20
4	41	110	0.37
5	40	86	0.47

the teams managed to divide the work between the sites in each team in such a way that each site could work independently and the 10 hour time difference between sites was not overly disruptive.

> *We didn't face any challenges [when working in a distributed team compared to a local team] because our tasks were modularized but, obviously if the tasks were interdependent, then we would have needed a lot more communication.*
>
> — Team Member, Canada

> *It's funny because they say communication is a problem in global, and this is true maybe. But in my experience, it's not, I did communicate better with my local team than I did with global team. [...] Which is true. But, I didn't see too much communication with my global team so necessary for the productivity to be successful. [...] Because we had, we split our tasks very well.*
>
> — Team Member, Canada

We saw evidence of this in the Flowdock communication logs. The cross-site teams had no more distributed communication within Flowdock than the local teams; 48% of Flowdock messages created by members of local teams were distributed communication, while 45% of the distributed team members messages were distributed. A Chi-Squared test of difference of proportions showed no difference in the breakdown of distributed and local communication when comparing distributed and local teams (x^2=2, p=0.16). Details in Table III.

TABLE III: Local vs. Distributed Communication

	Local Messages	Distributed Messages	Distributed Messages Ratio
Local teams	75	69	48%
Distributed teams	91	74	45%

The fact that the distributed teams worked well could also be seen in the teamwork quality measurement taken after each Sprint [26]. The means were 75.08 for distributed teams and 72.08 for local teams. Thus, interestingly, students reported teamwork quality to be a bit higher when working in distributed teams. This difference is not, however, statistically significant (Exact Wilcoxon signed rank test, p=0.07). Still, this is a very interesting finding as you could have expected the opposite: that the teamwork quality would have been clearly higher in local teams, when students can meet and work frequently face-to-face.

TABLE IV: Teamwork by site and distribution

Distribution	Teamwork quality	
	Finnish students	Canadian students
Local	66.73	75.11
Distributed	71.79	76.92

We also evaluated the STC by considering the local versus distributed communication within Flowdock. A coordination need was counted as local if both people involved in the coordination need were collocated. Otherwise, the coordination need was distributed. As shown in Table V, in all Sprints, the local STC scores are higher than the distributed STC scores. This indicates that when a coordination need exists between two developers who are collocated, it is more likely to be fulfilled. While, the difference in STC scores is only statistically significant for two of the four Sprints, the lack of significance in Sprint 5 is likely due to only the small dataset. Further, the local students are likely to have coordinated their work in other, non-traceable face-to-face communication. This shows that while the students did not report differences between local and distributed coordination, there is evidence that a difference did exist.

TABLE V: Sociotechnical Congruence in Distributed vs. Local Teams

Sprint	Local STC	Distributed STC	Chi-Squared Test
2	4/33=0.12	2/32=0.06	x2=0.67,p=0.41
3	40/107=0.37	4/113=0.04	x2=39.34,p<0.001
4	28/50=0.56	13/60=0.22	x2=13.75,p<0.001
5	22/38=0.58	18/48=0.38	x2=3.55,p=0.06

A few interviewed team members reported that after working in a global team, where they were vigilant about frequent communication, when they switched to a local team, they became too relaxed. They assumed within their local teams they would easily be aware of each others activities, and they did not distribute tasks well or communicate effectively. This resulted in duplicate work and reduced productivity.

> *Because we are local we expected that, you just work with it and.... so, story weren't split into tasks, people weren't being assigned immediately. We were like, oh these three people work on this story. [...] maybe because we thought we are co-located we are quite relaxed about, we are not so strict in being explicit on who is doing what. So we don't split and, that slows down productivity also.* — Team Member, Canada

Two of the teams worked in a local team during the first two non-training Sprints and were transitioned to global teams for the last two Sprints. We received positive feedback from them on this order, as taking good practices into use in a local team was seen easier, and thus the global team could benefit from the lessons learned during the local Sprints.

> *I think one of the disadvantages for most of those teams was, they've switched at the end. So they were local at the very end, whereas we switched at the beginning [...] So we, started working on agile, trying to get the practices in, and when we switched the Canadian team, we tried to merge our, team ideas of how to do things. [...] as we worked through that, things just got smoother and smoother, because it was on the Canadian local team. [...] because you had face-to-face communication [...] By the time we went back to the international teams, we had a better idea of, what works [...] So we were, better able to implement that with the international teams, which I think was really helpful.*
>
> — Team Member, Canada

The interviews indicate that there might be good reasons to start the project by local Sprints and then move on to a more challenging environment. Therefore, we believe that this is an interesting topic for further investigation.

Surprisingly, several students commented in the interviews that they actually preferred working in their distributed team than in their local team. They mentioned several reasons for that: work was better organized in a distributed team, the distributed team was smaller, communication and 'who was doing what' was more transparent in a distributed team as most communication was immediately available in Flowdock, while local teams sometimes waited until the next face-to-face

meeting, and finally the Finns were thanked for bringing up problems in a distributed team, whereas some students felt that local Canadian teams avoided facing problems.

> *I think I like the international team, a lot better. One, they are smaller. We had a lot better system for distributing our work. So we had [names of the team members] peer programming. Me and [name], would pair together and then [name] and [name]. [...] Having those interdependent three pairs, removed most of the dependencies, so it worked really well for us.*
>
> — Team Member, Canada

> *The funny thing was, particularly in my team, we worked better with the Finnish team than our own local team. [...] It could be because, Finnish people [...] are very straightforward. So, if they're silent we know that there is no problem. Or if there is a problem they used to speak. And we used to figure it out. But with Canadian local team, even if there's a big problem, nobody says anything. There's no meeting, nothing, no resolution. Sometimes it was way too polite and people don't say anything.*
>
> — Team Member, Canada

We examined how the teamwork quality developed over time. In Table VI, we can see that the teamwork quality improves slightly during the second Sprint for each team (Sprints 3 and 5), with the last Sprint being clearly the best. This result is expected, as the interviewed team members reported that during the first Sprint in a new team they learned to know each other and collaborate, while during the second Sprint they learned to work efficiently together.

TABLE VI: Teamwork quality over time

Sprint	Teamwork Quality
2	72.50
3	73.20
4	72.47
5	76.05

D. Inter-team Collaboration

As all the teams were building a common product and taking stories from the same backlog, inter-team collaboration was a crucial part of the work. The teams were expected to take care of collaboration and coordination between the teams by themselves, as in Scrum there are no project managers. According to the iteration surveys (Figure 3b) students were not satisfied with the amount of communication between their own team and other teams: almost half of the students felt that the amount of communication between the teams was too low.

In agile development, Scrum-of-Scrums [27] is used for inter-team coordination. Scrum-of-Scrums is a short status meeting with one representative from each team occurring daily or a few times per week. There, each team representative shares information from his or her own team by answering the three Scrum questions. After the meeting, each representative shares with their own team the relevant learnings.

We implemented a weekly Scrum-of-Scrums meeting during the course to share information and coordinate between the teams. Even though the teams found this practice useful, they had several challenges in implementing it: difficulties in knowing what kind of information would be of interest to the other teams and what might not be that interesting; the team representatives were not always well prepared for the meeting, i.e. they did not know well enough what their own team members had been doing; the team representatives did not always know which information would be interesting and important to report back to their team members, thus this reporting back was not always taken care of well; and finally, team representatives did not always participate in the meeting. Due to these challenges the Scrum-of-Scrum meetings were not as useful as they could have been. Some interviewees also mentioned the lack of inter-team communication as problematic. The quotes below highlight some of the challenges teams faced regarding Scrum-of-Scrums.

TABLE VII: Dependencies.

Sprint	Stories with Dependencies	Dependencies	
		Intra-team	Inter-team
2	41%	14%	86%
3	38%	23%	77%
4	36%	27%	73%
5	34%	52%	48%

> *And then you don't know that what information you need to share, what information you don't need to share. I think it [Scrum-of-Scrums] would have been more successful if everyone would have been there.*
>
> — Team Member, Canada

> *But people did not always show up. [...] and when they did show up they just told that our team has been doing this, I don't know anything about that, and I have done this. So the big picture was missing. [....] And when I followed the discussion [of some team] in the forum, it was possible to notice that the person [participating in Scrum-of-Scrums] had not distributed the information.*
>
> — Team Member, Finland

Despite the challenges with Scrum-of-Scrums meeting, the students considered it a useful practice and believed it should be continued and improved in the future: invite everybody to participate; adjust the time of the meeting to better accommodate the students; and, stress more in the beginning of the course why Scrum-of-Scrums are useful, how they should be organized, and how the students should prepare for them. The challenges the students reported are similar to findings in studies in industry [28].

E. Task Allocation

User stories were created by the Product Owner and self-selected by the teams. If dependencies exist between user stories, the teams responsible for those stories must coordinate their work, introducing additional coordination overhead. The PO created user stories with less dependencies over time, as shown in Table VII. However, dependencies between user stories cannot be fully eliminated. We, thus, analysed dependencies between stories to determine if teams were making smart choices when they selected their stories to minimize the number of dependencies across teams. For this analysis, we associated each story to the first Sprint in which it was worked since we were interested in studying the story selection. Thus, stories which were not completed in the Sprint in which they were initially selected, deferred stories, do not appear in multiple Sprints for this analysis.

We saw that the teams made better choices over time when self-assigning user stories, as the number of dependencies

within teams increased over time while the number of dependencies across teams decreased. In the interviews, several students explained that they selected user stories or tasks that were related to the stories or tasks that they worked during the previous Sprint. The students, therefore, likely gained a better understanding of the product's architecture throughout the course enabling them to better understand the dependencies that existed between the stories and make better decisions. Further, in the surveys, the students reported that the amount of communication between teams was too low. This was likely motivation for the students to eliminate the need to coordinate outside established team boundaries by reducing dependencies across teams.

F. The New Course Format

The students' reaction to this new course format, in which they worked in both local and global teams as opposed to only global teams, was positive. Regarding learning, the students were highly positive. The interviewed students felt they had reached or even exceeded their learning goals. They reported that the switch from global to local teams and vice versa helped them learn hands-on the differences between working in local and distributed teams.

> *You learn the difference pretty well. [...] I think the switch is educational. [...] Because it helps you to compare and contrast the differences between working in a local and working in a distributed team and what the challenges are for each. And we did have challenges in our local team too that were different from the distributed team. Such as getting together is in itself a big challenge.* — Team Member, Canada

However, the students were so committed to the project that most of our interviewees actually mentioned that when the teams changed, it slowed down their overall productivity and without the team changes they could have contributed even more to the project.

> *During the first Sprint we learned what this team can do, we learned who are in this team and at what level they can do work, or what kind of work they can do, and during the second Sprint we worked and then the new Sprint already started and we had to start that all over again. So it slowed down the pace quite a lot, before the new team could get started and we could figure out what this new team is capable of.* — Team Member, Finland

Overall, the student feedback on the course was highly positive: the students especially appreciated the possibility to work in a large, globally distributed project, for a real customer and for a product that is and will be used by real end-users, while at the same time learning the agile methods and practices that are successfully used in industry. They highlighted that learning Scrum, as well as the best practices for GSE projects, would not have been possible through theory and reading books alone. Instead, being able to use Scrum in practice, face GSE challenges in real-life and solve those challenges was a far better way to learn.

> *Practical experience is good. [...] I strongly believe the challenges was the spice in it. [...] Meeting obstacles and solving those obstacles was perfect learning curve in my opinion, and that's what made this course excellent.* — Team Member, Canada

The only significant change to the course the students would have made was to make it longer, so they could work for a longer period in the same team.

VI. Discussion

In this section, we summarize our results and answer the research questions.

RQ1 Did students experience differences in working in global vs. local teams, and if so, what?

Surprisingly, our results show that there were only non-significant differences between working in distributed versus non-distributed teams in terms of communication, teamwork and conformance to the used Scrum practices. Students were highly satisfied with the communication and Scrum practices in both types of teams. Regarding teamwork quality, surprisingly, distributed team members reported slightly, but not significantly, higher satisfaction in both the surveys and the interviews. We found these results surprising, since we would have expected to find clear differences between local and global teams, as it is well-known that globally distributed teams have many more challenges than local ones regarding communication and collaboration. The higher teamwork quality in global teams was particularly interesting, as it clearly contradicted our expectations since the global teams were unable to meet face-to-face and their synchronous communication was limited by a time difference of 10 hours. We suspect that the main reason for these findings, is that the Scrum process supported communication and collaboration in distributed teams extremely well, and thus helped alleviate many GSE challenges.

RQ2 Did students learn effective strategies for dealing with GSE challenges over time?

We were positively surprised by the fact that the student teams did not have big challenges in distributed collaboration. Besides learning the Scrum method and formal communication possibilities that it offers, our students learned effective informal communication strategies, e.g., they adopted Flowdock as the primary communication and coordination tool. Students reported that using this tool made it easier to stay aware of their teammates activities since all communication was in one repository.

Moreover, the students learned to divide work efficiently; our analysis showed that they made better choices over time in selecting stories to Sprints, as dependencies across teams decreased in later Sprints.

RQ3 How did the students perceive the new course format?

Students were highly satisfied with the course. Especially, they appreciated the possibility to work in a large, globally distributed project, for a real customer and for a product that is and will be used by real end-users, while at the same time learning the agile methods and practices that are currently becoming a standard in many companies.

Our new course format gave the students a possibility to practice working in both global and local teams, experience the challenges and try out good practices in both types of teams, as well as contrast their experiences. Student reaction to this new course format was positive. From the learning point of view the students were very satisfied with the new format, and encouraged us to keep it in the future, as they felt

they had reached and even exceeded the learning goals of the course. However, looking at the project from a productivity and efficiency point of view, the team switches in the middle of the project, of course, did not make sense. As the project was for a real customer the students felt that they could have contributed to the project even more without the switches. Thus, our students hoped that the future editions of the course would be longer, so that students would have more time working in each type of team, instead of having to switch teams right away when having just learned how to collaborate in that team and when the team starts to be productive.

VII. Conclusions

In this paper we presented an new course format for teaching students global software engineering using Scrum practices adapted to a globally distributed environment. The new element of this course was the innovative team set-up that provided all students a possibility to work both in a local and a global team, and thus contrast their experiences from both.

We studied the teams and student learning using a mixed-method approach. Surprisingly, our main finding was that the students did not report significant differences between working in local versus global teams in regards to communication, teamwork and conformance to Scrum practices used. This result contradicts clearly with our previous knowledge on GSE, as globally distributed teams normally face many more challenges than local teams especially regarding communication and collaboration. The main explanation to this finding seems to be the Scrum method that helps alleviate many GSE challenges. We feel that this finding would be an interesting and important topic for further studies.

Overall, our students were highly satisfied with the course and with working in both local and global teams. Based on our experience we warmly encourage other teachers to try out a similar course format: even though this format needs a lot of preparation from the instructors, based on the student reaction and learning, it provided excellent results.

References

[1] E. Hossain, M. A. Babar, and H.-y. Paik, "Using scrum in global soft. development: A systematic literature review," in *Proc. of the 4th IEEE Intl. Conf. on Global Soft. Eng.*, 2009, pp. 175–184.

[2] G. Hanssen, D. Smite, and N. Moe, "Signs of agile trends in global soft. eng. research: A tertiary study," in *Intl. Conf. on Global Soft. Eng. Workshop (ICGSEW)*, 2011, pp. 17 –23.

[3] M. Paasivaara, S. Durasiewicz, and C. Lassenius, "Using Scrum in a globally distributed project: A case study," *Soft. Process Improvement and Practice*, vol. 13, no. 6, pp. 527–544, 2008.

[4] C. Scharff, O. Gotel, and V. Kulkarni, "Transitioning to distributed development in students' global soft. development projects: The role of agile methodologies and end-to-end tooling," in *5th Intl. Conf. on Soft. Eng. Advances (ICSEA)*, 2010, pp. 388 –394.

[5] C. Scharff, "Guiding global software development projects using scrum and agile with quality assurance," in *24th IEEE-CS Conf. on Soft. Eng. Education and Training (CSEET)*, 2011, pp. 274 –283.

[6] C. Scharff, S. Heng, and V. Kulkarni, "On the difficulties for students to adhere to scrum on global soft. development projects: Preliminary results," in *Collaborative Teaching of Globally Distributed Soft. Development Workshop (CTGDSD)*, 2012, pp. 25 –29.

[7] M. Paasivaara, C. Lassenius, D. Damian, P. Räty, and A. Schröter, "Teaching students global software engineering skills using distributed scrum," in *Companion Proceedings of 35th International Conference on Software Engineering, ICSE 2013*, 2013, pp. 1128–1137.

[8] K. Schwaber and M. Beedle, *Agile Software development with Scrum*. Prentice-Hall, 2002.

[9] L. Fortaleza, T. Conte, S. Marczak, and R. Prikladnicki, "Towards a gse intl. teaching network: Mapping global soft. eng. courses," in *Collaborative Teaching of Globally Distributed Soft. Development Workshop (CTGDSD), 2012*, june 2012, pp. 1 –5.

[10] A. Mockus and J. D. Herbsleb, "Challenges of global software development," *7th Intl. Symp. on Soft. METRICS*, pp. 182–184, 2001.

[11] M. Monasor, A. Vizcaíno, M. Piattini, and I. Caballero, "Preparing students and engineers for global soft. development: A systematic review," in *Intl. Conf. on Global Soft. Eng.*, 2010, pp. 177 –186.

[12] I. Richardson, S. Moore, D. Paulish, V. Casey, and D. Zage, "Globalizing software development in the local classroom," in *20th Conf. on Soft. Eng. Education Training (CSEET)*, 2007, pp. 64 –71.

[13] K. Swigger, R. Brazile, F. Serce, G. Dafoulas, F. Alpaslan, and V. Lopez, "The challenges of teaching students how to work in global soft. teams," in *Transforming Eng. Education: Creating Interdisciplinary Skills for Complex Global Environments*, 2010, pp. 1 –30.

[14] Y. Cai and W. Baelen, "On the development of pedagogical materials for globally distributed software engineering," in *Proc of Collaborative Teaching of Globally Distributed Software Development - Community Building Workshop*, 2012.

[15] J. D. Herbsleb and R. E. Grinter, "Splitting the Organization and Integrating the Code: Conway's Law Revisited," in *Proceedings of the 21st International Conference on Software Engineering*. New York, NY, USA: ACM, 1999, pp. 85–95.

[16] M. Cataldo and J. D. Herbsleb, "Coordination breakdowns and their impact on development productivity and software failures," *Software Engineering, IEEE Transactions on*, vol. 39, no. 3, pp. 343–360, 2013.

[17] D. Damian, L. Izquierdo, J. Singer, and I. Kwan, "Awareness in the wild: Why communication breakdowns occur," in *Global Software Engineering, 2007. ICGSE 2007. Second IEEE International Conference on*. IEEE, 2007, pp. 81–90.

[18] C. R. de Souza and D. F. Redmiles, "An empirical study of software developers' management of dependencies and changes," in *Proceedings of the 30th international conference on Software engineering*. ACM, 2008, pp. 241–250.

[19] C. R. de Souza, S. Quirk, E. Trainer, and D. F. Redmiles, "Supporting collaborative software development through the visualization of socio-technical dependencies," in *Proceedings of the 2007 international ACM conference on Supporting group work*. ACM, 2007, pp. 147–156.

[20] K. Schwaber and J. Sutherland, "The scrum guide — the definitive guide to scrum: The rules of the game," www.scrumguides.org, 2013.

[21] M. Paasivaara and C. Lassenius, "Collaboration practices in global inter-organizational soft. development projects," *Soft. Process Improvement and Practice*, no. 4 (8), pp. 183–199, 2003.

[22] T. Jick, "Mixing qualitative and quantitative methods: Triangulation in action," *Administrative Science Quarterly*, vol. 24 (4), 1979.

[23] H. Gall, K. Hajek, and M. Jazayeri, "Detection of logical coupling based on product release history," in *Proceedings of the International Conference on Software Maintenance*, ser. ICSM '98. Washington, DC, USA: IEEE Computer Society, 1998, pp. 190–. [Online]. Available: http://dl.acm.org/citation.cfm?id=850947.853338

[24] G. Spreitzer and A. Mishra, "Giving up control without losing control - Trust and its substitutes' effects on managers' involving employees in decision making," *Group & Organization Management*, vol. 24 (2), pp. 155–187, 1999.

[25] M. Hoegl and H. Gemuenden, "Teamwork quality and the success of innovative projects: A theoretical concept and empirical evidence," *Organization Science*, vol. 12 (4), pp. 435–449, 2001.

[26] M. Hoegl, K. Weinkauf, and H. Gemuenden, "Interteam coordination, project commitment, and teamwork in multiteam R&D projects: A longitudinal study," *Organization Science*, vol. 15 (1), pp. 38–55, 2004.

[27] J. Sutherland, "Agile can scale: Inventing and reinventing scrum in five companies," *Cutter IT Journal*, vol. 14, no. 12, pp. 5–11, 2011.

[28] M. Paasivaara, C. Lassenius, and V. Heikkila, "Inter-team coordination in large-scale globally distributed scrum: Do scrum-of-scrums really work?" in *Empirical Software Engineering and Measurement (ESEM), 2012 ACM-IEEE International Symposium on*, Sept 2012, pp. 235–238.