IEICE TRANS. INF. & SYST., VOL.E105–D, NO.1 JANUARY 2022

PAPER Special Section on Empirical Software Engineering

Balanced, Unbalances, and One-Sided Distributed Teams - An Empirical View on Global Software Engineering Education

Daniel Moritz MARUTSCHKE[†a)], ***Member***, **Victor V. KRYSSANOV**[††b)], ***and*** **Patricia BROCKMANN**[†††c)], ***Nonmembers***

SUMMARY Global software engineering education faces unique challenges to reflect as close as possible real-world distributed team development in various forms. The complex nature of planning, collaborating, and upholding partnerships present administrative difficulties on top of budgetary constrains. These lead to limited opportunities for students to gain international experiences and for researchers to propagate educational and practical insights. This paper presents an empirical view on three different course structures conducted by the same research and educational team over a four-year time span. The courses were managed in Japan and Germany, facing cultural challenges, time-zone differences, language barriers, heterogeneous and homogeneous team structures, amongst others. Three semesters were carried out before and one during the Covid-19 pandemic. Implications for a recent focus on online education for software engineering education and future directions are discussed. As administrational and institutional differences typically do not guarantee the same number of students on all sides, distributed teams can be 1. balanced, where the number of students on one side is less than double the other, 2. unbalanced, where the number of students on one side is significantly larger than double the other, or 3. one-sided, where one side lacks students altogether. An approach for each of these three course structures is presented and discussed. Empirical analyses and reoccurring patterns in global software engineering education are reported. In the most recent three global software engineering classes, students were surveyed at the beginning and the end of the semester. The questionnaires ask students to rank how impactful they perceive factors related to global software development such as cultural aspects, team structure, language, and interaction. Results of the shift in mean perception are compared and discussed for each of the three team structures.

key words: *global software engineering, online education, cultural dimensions, distributed development, project based*

1. Introduction

In today's software development landscape, projects are often conducted by international teams distributed in different countries around the world. In Global Software Engineering (GSE), interconnectedness and tools to facilitate asynchronous working have driven this trend. Differences in expertise, but also diverging wages are factors that lead to team members in different countries being assigned specific roles. Roles based on location then result in offshoring projects. Requirements engineering tasks are often assigned to an onsite group with direct access to customers, whereas development tasks can be outsourced to geographically distant groups residing in other countries. Main influencing factors for offshoring IT has been examined by Agrawal et al. [1]. The trend of offshoring is discussed and they found that American organizations plan to increase these activities in the future. Agrawal et al. recommend an inclusion of offshoring activities in IT curricula.

Culture, and its difference between distributed software development teams, can be a major source for misunderstanding. Hofstede et al. has written extensively about differences in cultural dimension, such as power distance, individualism vs. collectivism, uncertainty avoidance, long-term vs. short-term orientation, indulgence vs. restraint, and assertiveness vs. cooperation (the latter referred by Hofstede as masculinity vs. femininity) [2].

Hall and Hall published work that introduced the concept of high-context and low-context cultures [3]. Western cultures have tendencies towards low-context culture, whereas Asian cultures tend to be high-context. In low-context cultures, such as in Germany, communication is conducted more direct and verbose. Words explicitly written and spoken are to be taken literally. In high-context cultures, such as Japan, non-verbal cues including gestures, body language, and interpersonal relationships have to be taken into consideration and can be more important than written or spoken words.

Unique circumstances for the instructors of GSE education have to be expected for every new course. One factor that can widely vary is the number of students enrolled on each side. While some aspects can be compensated, careful planning at the beginning of a project is necessary to ensure adequate learning outcomes [4], [5].

The benefit of empirical studies that emphasize the pedagogical challenges and values are formulated by Carver et al. in cost and benefit analyses with strategies to fully utilize these studies [6].

This paper reflects on three different team structures of global software engineering education over a period of four years between a Japanese and German university. An empirical analysis of lessons learned and best practices are presented for these team structures: *balanced*, where the number of students either in Germany and Japan is not more than

Manuscript received February 26, 2021.
Manuscript revised July 14, 2021.
Manuscript publicized September 30, 2021.

[†]The author is with the Ritsumeikan University College of Global Liberal Arts, Ibaraki-shi, 567–8570 Japan.
[††]The author is with the Ritsumeikan University College of Information Science and Engineering, Kusatsu-shi, 525–8577 Japan.
[†††]The author is with the Nuremberg Institute of Technology, Keßlerplatz 12, 90489 Nuremberg, Germany.
a) E-mail: moritz@fc.ritsumei.ac.jp
b) E-mail: kvvictor@is.ritsumei.ac.jp
c) E-mail: patricia.brockmann@th-nuernberg.de
DOI: 10.1587/transinf.2021MPP0002

double their counterpart, *unbalanced*, where the number of students on one side is more than double their counterpart, and *one-sided*, where there were no students on one side.

The rest of the paper is structured in five parts. The following Sect. 2 covers previous works. Section 3 describes the framework, in which the global software engineering education was conducted with detailing learning goals (Sect. 3.1), CDIO (Conceive-Design-Implement-Operate) guidelines (Sect. 3.2), and project-based course structure (Sect. 3.3). An empirical analysis of three different types of team structures are provided in Sect. 5, with the balanced team structure in Sect. 5.1, the unbalanced team structure in Sect. 5.2, and the one-sided team structure in Sect. 5.3. Results of the three team structures, reoccurring experiences, and student surveys and insights gathered from their perceived importance of GSE factors are reported in Sect. 6. The paper concludes in Sect. 7 and lays out future implications.

2. Related Work

Global software engineering education posed many difficulties and has been the research focus for a number of authors. Beecham et al. have written a comprehensive literature review on 82 articles to identify major challenges in GSE education [7]. Many of these challenges are also faced in real-world projects, where others are unique to teaching distributed software development: global distance, teamwork, soft issues, stakeholder roles, infrastructure, the development process, curriculum, and pedagogy. Schneider et al. reviewed 330 papers to map out strategies and solutions for GSE in the industry [8].

A study by Paasivaara et al. discusses the experience with a cooperative course taught by universities in Finland and Canada, where students had the opportunity to work with single-site and cross-site teams [9]. Here, no significant differences were found between the two team structures. Their suggestion is the use of Scrum bridging the gaps in team differences.

When older and younger students are mixed in self-organizing teams, older students preferred to divide project tasks based on architecture and to form co-located groups. Younger students on the other hand tended to form cross-site teams on their perceived abilities, leading to an improved project performance. Bosnić and Čavrak laid out their findings, underlining the importance of teaching students to work in cross-site teams [10]. Further investigations by Bosnić et al. focused on analyzing the impact of diversity on three aspects of distributed course organization: institutions, teaching, and projects. As one of their recommendations is to develop a common baseline vie MoUs (Memorandum of Understanding) to deal with institutional differences, such as number of credit points, grading, course structure, and others. They also issue a caution for the teaching side, that instructors would have to expect varying circumstances that would need different approaches every year [4]. With this, they acknowledge that inter-university cooperative courses must often expect and accept the fact that a different number of students would take GSE courses each year. Where the number is not equal, they experienced low project success and had to re-run project proposals and were not able to adequately support many of the learning goals.

Different instruction methods were classified by Clear and Beecham with their continuum model of GSE education [11]. Fully immersive courses, often based on project-based learning, are classified as the most demanding on both the instructors and students. Well implemented distributed courses that involve two ore more universities' collaboration are placed towards the top of the continuum. Other classes with lower efforts, such as case studies, exercises, role-play, and simulations can be conducted as single courses in the classroom.

Hoda et al. stressed the importance of learning to deal with diverse languages, concepts of time, and assumptions about culture [12]. How the nature of time can be perceived differently is described by Hall with the M-time—time as a series of distinct, monochromatic units—and P-time—time as fluid and polychromatic [3]. Hoda et el. define seven dimensions of socio-cultural distance, causing the most significant changes in their course: language differences, concept of time, attitude towards grades, assumptions about national culture, differences in autonomy, influence of the course lecturer, and work habits.

3. Educational Framework of Global Software Engineering

How knowledge and experience are managed in GSE education require careful considerations. Learning goals and learning environment have to match for a successful outcome [7], [8].

This paper's research is motivated by (global) software engineering education generally lagging behind software development practices. Global software engineering education is still uncommon in higher education, as it requires more work from all sides, traditional class structures are largely irrelevant, and most attempts fail. The research objective is to investigate preferred formal class organizations for global software engineering education.

Students have to understand key problems in distributed software system development. They have to handle tools for distributed collaboration, such as cloud platforms, video conferencing software, agile tools, such as Jira. Technical knowledge must be experienced, such as universally understood concepts (UML, state-diagram, flow-chart, etc.), good programming practices, and modern practices of software engineering. Management methods for distributed project groups and distributed agile methods have to be worked with. On an intercultural level, communication with project members from different countries must be faced. All has to be operated in an ethical scheme to foster team communication, information exchange, and respect.

3.1 Learning Goals

There are two learning objectives for global software engineering: 1. increasing knowledge about technical skills related to GSE and 2. honing non-technical skills related to intercultural collaboration.

As described by Colomo-Palacios et al., students of GSE education should learn a number of technical skills: requirements engineering, engineering design (UML, state-diagram, etc.), software construction, testing maintenance, configuration management, quality management, tools and methods, and software engineering process models, such as phase-based or agile development models [13]. While these build a strong foundation for software engineering, changing industry needs have to be accommodated in regular curricula updates. Technical skills are a necessary foundation for a career in software engineering, Joseph et al. claim that these skills alone are not sufficient for success in an IT career [14]. Thus, even with covering technical skills, the need for software engineering, and especially for global software engineering are not met. A balanced strategy is needed to put into action a merged approach of technical and non-technical skills.

Identifying non-technical skills may depend on the cultures involved and the environment a project is being organized. Several authors have conducted studies on this topic to determine the skills that students should master for global software engineering. The importance of controlling geographical distance, collaboration and learning to work together as a distributed team, and coordinating distributed software development processes are stressed by Beecham et al. and Clear et al. [5], [7]. A shift from acquiring knowledge to the mastery of skills might be necessary in an increasingly globalized working landscape and for sustainable development. Damasevicius et al. affirm this from a long-term view on learning [15].

3.2 CDIO Guidelines

To have a guided approach to software engineering, the GSE course is structured following the Conceive, Design, Implement, and Operate (CDIO) stages [16]–[18]. To be able to cover all aspects of a real-world development scenario, an overlap of around 12 weeks with the collaborating universities is necessary. To adopt these stages effectively, the overlapping weeks are divided into design phase and prototype phase. This helps students to finish either with a woking prototype or a concept demonstration. Topics that are covered include: software lifecycle and its models; quality management, process improvement techniques in virtual teams and distributed projects; modern practices and future trends in software development; socio-technical systems, outsourcing and global software development; advanced techniques of requirement elicitation; software project management, modern approaches to management, risks and risk management in software development projects; advanced techniques of software development, i.e., software reuse, reference architecture, open source software; software testing and validation, modern approaches to software testing and certification; software product documentation, software documenting tools, Unified Modeling Language (UML).

The importance of requirements engineering and validation are stressed by Crawley et al. [19]. Where requirements engineering and updating specifications are done throughout the semester, the validation is done during the students final presentation or final written report.

3.3 Project-Based Learning

The role of student-centric teaching has been growing in engineering education. With the need to resemble scenarios that are close to experiences in the industry, concepts like problem-based and project-based learning resurfaced in software engineering education. Although the two terms are often used interchangeably, there are significant differences. Savery frames Project-Based Learning as being externally defined as the desired end product, rather than the Problem-Based Learning, which allows the students to define project specifications intrinsically. As software engineering project descriptions and constraints are usually defined by the customer, the GSE classes were taught using Project-Based Learning. The use of this format has been reported to be highly effective in teaching global software engineering [7], [8], [20]–[23].

The learning success of using a project-based learning environment for GSE education depends on well implemented guidelines by instructors. A disciplined planning phase of instructors from all sides is required, as well as a high degree of guidance and oversight throughout the project.

Students were typically assigned to work on a concrete project for a real-world customer. The four projects that are described in this paper were a customer loyalty system for an Irish pub, an e-voting system prototype, a university laboratory scheduling and management system, and an elevator scheduling and optimization system. While the frameworks for each were provided by the instructors, specifications and requirements had to be identified, defined, and analyzed by the students.

4. Research Method

Factors that students evaluated by their perceived influence on global software engineering in 2017/18 are listed below. The questionnaire was handed out after the semester project had concluded. The following seven factors for the questionnaire are based on the extensive research of Hofstede et al. [2].

Initial factors: *Geographical distance*, *Time zone difference*, *Language differences*, *Proficiency in shared language*, *Cultural differences*, *Familiarity between teams*, *Trust between teams*.

Based on the instructors' previous research and com-

bined experiences of more than 20 years in teaching global software engineering, factors added to the list from 2018/19 onward are as follows.

Added factors: *Transparency and accountability*, *Communication between teams*, *Software and hardware tools*, *Leadership*.

All 11 factors were surveyed with an ex-ante (before the semester) and ex-post (after the semester) questionnaire.

The questionnaire style in 2017/18 and 2018/19 was adopted from previous research on the German side and asked students to rank factors by their importance (from one to seven in the former and from one to 11 in the latter). This lead to occasional confusion or unusable results, as a rank was misunderstood as a scale, resulting in duplicate or skipped ranks. In 2019/20 and 2020/21, the questionnaire was redesigned with a five-step Likert scale for each factor, ranging from *not impactful* to *very impactful*.

The ex-ante and ex-post approaches enable to also observe a shift in perception from expectations before the project and an experience report after completion.

The process model follows the software reuse development, where students assembled working prototypes mainly from existing components. For reference, other popular software life cycle models like the waterfall model, evolutionary development, and formal system development are covered in theory.

5. Empirical Analysis of Team Structures

The global software engineering course taught at the Nuremberg Institute of Technology in Germany and at Ritsumeikan University in Japan has been successfully conducted for four years. Within this period, three different course structures emerged out of necessity due to varying number of students on both side. The first two years were experienced in a balanced way, where the number of students was similar on each sides. In 2019/20, the numbers were highly unbalanced with 28 students in Germany and 4 students in Japan. In 2020/21, there was again a large number of students in Germany, but no students in Japan, resulting in a one-sided team structure. Table 1 shows the numbers for each year and the resulting team structure. It is to note that students in Germany tend to be more homogeneous (mostly German) and the students in Japan heterogeneous (from Asian countries, typically China, Korea, Vietnam, and Japan) as part of international master's program. Students are part of the Information Science and Engineering department in Japan and the Computer Science department in Germany.

During the years 2017/18 to 2019/20 with students enrolled in both universities, one instructor in Japan gave weekly mini-lectures, participated by all members in Germany and Japan. The rest of the 90-minute class was used for communications between team members, with instructors on both sides observing, giving support and standing by for questions. In 2020/21, the lesson time was used to communicate with the customer, whose role the instructor in Japan took over. A second instructor in Japan joined on a voluntary basis and kept lesson minutes (protocol) and gathered insights from the Japanese and German side regarding the GSE projects.

Table 1 Number of students by year and resulting team structure.

	2017/18	2018/19	2019/20	2020/21
Germany	7	9	28	20
Japan	7	5	4	0
Structure	Balanced	Balanced	Unbalanced	One-sided

Through the four years, all students in Germany and Japan successfully finished the course. One student had to withdraw during the Covid-19 pandemic.

In the years except for 2019/20, students met weekly with the client during class (see Sects. 5.1 and 5.3). All customers thus far were located in Japan. In 2019/20, only students in Japan had access to the clients (university professors leading a laboratory), as they were assigned the role of the requirements engineering office (see Sect. 5.2). Students would meet at least once a week with clients, in some cases more than once after consulting with the development team in Germany.

The GSE courses were conducted in the winter semesters (fall semester in Japan) of 2017/18 to 2020/21. This section describes the different projects, their characteristics, and the empirical findings from the different experiences.

5.1 Balanced Team Structure

In 2017/18, students were assigned to develop a prototype of a loyalty system for an Irish pub operating in Japan. Two competing teams opted for different approached to this task. One team took a future-oriented route and developed a prototype programmed in Python based on the Microsoft Azure's platform. The other team chose the reliability of proven technology and demonstrated a simple, reliable, and low cost barcode-based system running on smart-phones. Difficulties in the beginning arose from communicating with the client and correctly understanding her needs and wishes. One obstacle was identified as a lack of dialogue on a non-technical plane. Another was the difficulty in understanding, i.e., translating the client's propositions into the technical domain. It was challenging for students to take lectures on cultural dimensions seriously, as they might feel more comfortable with problems they can assert some control over [2].

The following year 2018/19, students developed a concept for an e-voting system. Two cross-site teams were formed, each composed of two/three members from Japan and five/four members from Germany, respectively. Each team worked on developing their own prototype solution, which was presented to an independent customer at the end of the semester. One team developed a low-cost prototype which was based on a combination of analog mail, a personal ID card and asymmetric cryptography for user authentication. The second group developed a prototype based on innovative blockchain technology. Students ini-

tially had quite a bit of difficulty communicating with their remote counterparts in the other country. Challenges which could be anticipated, such as time differences between Japan and Germany, were handled quite well. Language difficulties, although also expected, were not so easily overcome. Switching from verbal communication to text chats greatly improved understanding between team members in difference countries.

5.2 Unbalanced Team Structure

In 2019/20, a scheduling and management system for a Japanese university laboratory was prototyped. The number of students was seven times higher in Germany than in Japan, which made a separation into evenly spread-out teams unfeasible. Instead, students in Japan were assigned the role of a requirements engineering office (team) that consulted with four different development teams in Germany. To even out the responsibility, each of the students in Japan was made accountable for a team in Germany. German students initially expressed their unhappiness to assume the "inferior" role, as they were confronted with their own stereotypes that Germany, as an industrialized country, should assume the role of the requirements engineers. This sentiment was reported similarly by Vallon et al., where the Indian team would have preferred to play the role of the developers rather than the customers [24].

5.3 One-Sided Team Structure

The latest project in 2020/21 was the development of an elevator optimization platform that allowed to adapt to different circumstances, such as capacity of a building, time of day, elevator count, and a Covid-19-mode. Goals were defined as minimizing waiting time, minimizing the number of occupants, all without disproportionately increasing the energy consumption.

As the course lacked the minimum number of students on the Japanese side, the instructors faced a one-sided team structure with 20 students on the German side and zero in Japan. Instructors in Japan took over the role of the customer, with three teams in Germany.

6. Results

(1) Balanced team structure:

In terms of the teaching experiences, these two did not provide significant insights. Balanced team structures are most common and there are many theoretical frameworks to depend on as well as ample empirical research. Each new project, however, comes with its unique dynamics and instructors have to adjust lecturing and advising students accordingly.

(2) Unbalanced team structure:

Early on, students realized that both the German and Japanese sides worked with different contextual backgrounds. This became apparent when students in Japan referred to elements commonly known about Japanese universities' laboratory environments, i.e., students being assigned to a laboratory with one leading professor and other instructors. Regular seminars, meetings, and students' responsibility to study at their designated desks were factors to consider in the prototype.

A unique aspect noticeable was the competitiveness of different teams leading to secrecy between German teams.

A successful GSE class could be realized by assigning the side with a smaller student number the role of a requirements engineering team and divide the larger side into teams with 7 ± 2 students. Research shows a correlation between team size and productivity and suggest more than three members for increased performance with teams larger than nine diminishing in productivity [25], [26].

(3) One-sided team structure:

Due to the Covid-19 pandemic, students in Germany were forced to participate remotely. This seemed to have advantageous effects on organization and productivity. This could be a consideration for future GSE classes. Even with face-to-face classes, artificially imposed restrictions to online participation could have similar positive effects. On the other hand, students were already used to online learning environment from the summer semester. Without this routine and the accompanying technological advances, this format would have been difficult.

In this class, students with experience in software development and agile developments, such as Scrum from work. Four to five students were the main drivers and took decisive leadership to help and compensate for weaker participants. Students were also more homogeneous, which removed language difficulties between on-site teams. Students formed three teams with different objective to contribute to the final product: Controller, Simulation, and Traffic Generation. The elevator optimization solution was implemented by using genetic algorithm, simulated annealing, and greedy algorithm with the option for the user to compare and choose.

One effect that could be observed was that presentations lacked a structured science and engineering style format. This could be explained by the lack of a team in Japan. The laboratory and seminar style in Japanese universities train students in consistent engineering presentations.

6.1 Reoccurring Experiences

There are certain themes that the instructors of the German-Japanese GSE courses could observe over the years. Dynamics of group forming was mostly independent of the culture and project could be observed on each side where teams had to work together. According to Tuckman, teams work through four stages: forming, storming, norming, and performing [27]. Students are initially unsure of the team's and project's purpose and they need to form their team identity.

In the storming phase, members tend to establish their perceived role, which can lead to conflicts. With time, teams arrange themselves when members resolve conflicts and find their roles. In the last stage, productivity is visible by most students working together towards a common goal. In the case of the four GSE courses, teams took typically three to four joint weeks to realize that only persistent work leads to an adequate result and students significantly intensified their efforts. The German teams usually work very task-oriented from the beginning of a project. A very direct communication style can be viewed as impolite by collectivist East Asian societies [2]. To mend the rift in team harmony, team building time was necessary in the beginning phase.

Taking team dynamics into account, constructive but decisive criticism around the middle of the semester as well as at the end of the semester seemed to not only boost the project outcome, but also increase the involvement of students otherwise operating in the background. Prolonged break periods should also be taken into account. In the case of the winter semester, the Christmas and winter break can pose a hazard for performance decline.

Pitfalls of previous projects are discussed with students at the beginning of the semester. Many of the same issues, however, present themselves during the semester, which were found to be a good way for students to connect the theoretical lecture with actually facing a problem.

A peculiar reoccurrence was seen in the first three years, where in the first third of the semester, students on each side voiced concern about the respective distributed team's English proficiency. The instructors consult at this stage with the students to point out ways to bridge language discrepancies by incorporating universal technical language and diagrams.

With an individualistic low-context culture in Germany and a collectivist high-context culture in Japan, the power distance between professors and students had different impact on students. A low power distance in Germany and a high power distance in Japan had to be compensated by the respective instructor. With this experience, students expressed the opinion at the end of the semester, that the distributed, project-based class more closely reflected a real-world scenario.

The scenarios described in this paper and according to Bosnič's findings, instructors need to be prepared to find flexible solutions to unforeseen problems [4]. While the different team structures covered in this research can be applied as a framework solution, a multitude of minor changes need to be addressed with each project. The group dynamic of students also requires individual attention, as does the inevitable evolution of this dynamic throughout the semester.

Different institutions follow slightly different semester and academic year schedules. For a successful outcome of global software engineering education, an overlap of 12 weeks has been proven to be adequate. An in-person meeting of instructors or the introduction via a dependable contact greatly solidifies trust in the establishment of a GSE collaboration. Planning for a new (but based on continued collaboration) GSE class should be planned towards the end of the previous semester.

Table 2 Participation rate of the ex-ante and ex-post questionnaires.

	2018/19	2019/20	2020/21
Germany before	9	25	9
Germany after	9	20	8
Japan before	5	4	Na
Japan after	5	4	Na

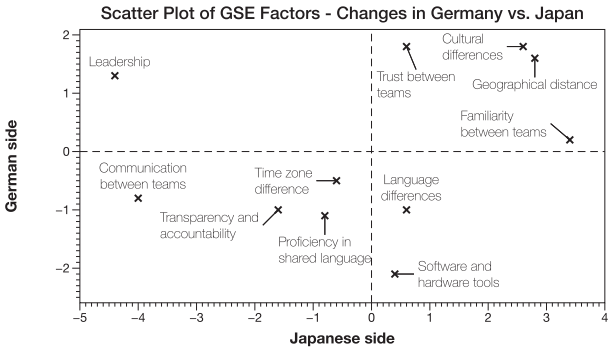

Fig. 1 Scatter plot of the shifts from before to after the GSE project in the 2018/19 semester.

6.2 Survey Results

Students were surveyed using anonymous questionnaires to better understand their observation and experience of GSE related factors.

The participation rate is detailed in Table 2, contrasting the number of students before the start of the project and after completion in Germany and Japan. Figures 1–3 visualize the perceptual shift of students from before starting the GSE project to after completion. Values are the differences of the means of the ex-ante and ex-post questionnaire results. Quadrant 1 indicates factors that have risen in importance for both the German as well as the Japanese side. Quadrant 2 shows an increase on the German side, but a decrease on the Japanese side. Quadrant 3 points to a perceptual decrease of factors for both German and Japan-based students. Quadrant 4 indicates an increase on the Japanese side, but a decrease on the German side.

Nine students from Germany and five students from Japan participated in both the ex-ante and ex-post questionnaires of 2018/19. Results are shown in Fig. 1. Due to a ranking of the factors, numerical values ranged from 1 to 11. The biggest absolute change on the German side was for software and hardware tools. Students were enthusiastic about tools in the beginning of the semester and hoped to overcome geographic or language barriers with these. On the Japanese side, the largest two factors were familiarity between teams, which experienced the largest positive change, and leadership, which dropped in importance the most. The biggest joint increase could be observed in cultural differences and geographic distance. Students expressed the opinion that working in a cross-site teams de-

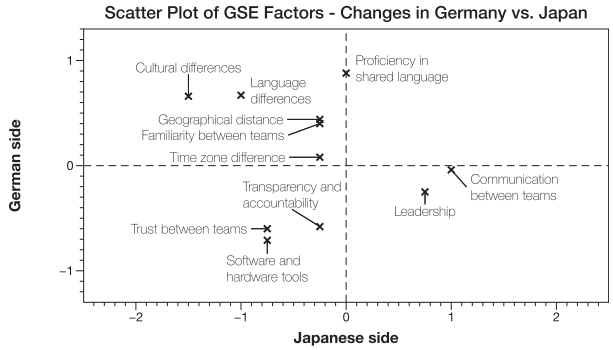

Fig. 2 Scatter plot of the shifts from before to after the GSE project in the 2019/20 semester.

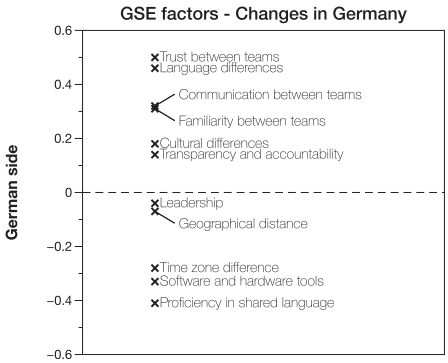

Fig. 3 Shift from before to after the GSE project in the 2020/21 semester visualized.

creased the amount of communication necessary. Students communicated synchronous during lecture hours and asynchronously during the week. Adverse effects could be noticed by students during the semester and are factors to look out for. Students typically reported that they valued experiencing cultural differences as a better learning experience than instructor-based lectures.

In 2019/20, 25 students filled out the ex-ante questionnaire, but this number dropped to 20 students for the ex-post questionnaire. All four students in Japan took both questionnaires. Results are shown in Fig. 2. The questionnaires had been changed to incorporate a five-point Likert scale with 1: not impactful to 5: very impactful. While no factor fell into quadrant 1 (positive for German and Japan-based students), the largest common decrease could be noted in software and hardware tools and trust between teams. The hope for reliance on technical tools is a common thread, where students realize there is no shortcut for joint work. While trust between teams is known to be an important factor in software development, the decrease could point to a successful, yet stressful project. For the students in Japan, communication between teams and leadership was deemed more important in retrospect, with a slight to negligible decrease on the German side. Proficiency in shared language had a significant increase for German students, where there was no shift on the Japanese side. Two factors had the German and Japanese side shift in opposite directions: cultural differences and language differences. German students viewed this more impactful at the end of the semester, students in Japan perceived it significantly less important. This indicates the homogeneous teams in Germany with less day-to-day language challenges and more homogeneous teams in Japan, with regular encounter with cultural and language barriers. The discrepancy between Fig. 1 and Fig. 2 could be explained by the different team structures or heterogeneity in participants and needs further investigation.

The number of participants in both the before and after questionnaire was lower, with nine in the beginning and eight in the end of the semester 2020/21. As all forms of contact was done remotely, the lower psychological pressure of physical attendance is a likely explanation for this. Results are shown in Fig. 3. The results are numerically comparable with the previous year and for that reason kept in similar graphical form. Interpretations for the concept of *teams*, however, should be viewed according to the one-sided team structure. Students formed three collaborating development teams, which operated remotely (distributed), but in the same timezone and cultural context. Other GSE factors can be viewed as related to communication with the customer in Japan. Trust between teams, communication between teams, and familiarity between teams were considered more impactful after the completion of the project, which denotes increased cohesion and performance of the teams at the end of the semester. The difference in language had a positive shift in importance, but the proficiency in shared language experienced a drop. Using a non-native language to work on projects influenced the GSE collaboration more than originally imagined, whereas the actual proficiency was deemed less important than expected. The use of presentations and demonstrations could explain this dynamic.

7. Conclusions

A number of conclusions can be drawn from the empirical reports of three different distributed team structures. Virtual, collaborative courses distributed in two countries with varying structures are a realistic alternative to teach global software engineering. However, instructors have to be prepared to find flexible solutions to unanticipated complications. Long-term collaborations with distributed teams should be planned with institutional rules of partnering universities in advance. Students should have adequate time throughout the semester to work on a project with their cross-site partners. At least 12 weeks that overlap with 90 minutes each week give students the necessary conditions to collaborate on large projects in a meaningful way. Differing credit points at each institution can be compensated by allowing the local instructor at each university to follow their grading criteria and assign grades to their own students.

A project-based approach in combination with a well organized curriculum is necessary for successful global software engineering education. Constant guidance and feedback to students by the instructors keep distributed

projects on track. Regular emphasis on universally understood concepts, such as good programming practices, UML, flowcharts, diagrams, and others keep students on track for a project.

Cultural differences, such as high or low power distance have to be taken into consideration, especially when projects include students from individualistic and collectivist cultures. To make trust a conscious subject matter, team building and trust exercises can be implemented in the beginning of the semester. This can help to keep the performance on a high level.

The communication between instructors is a key component for successful long-term collaborations. Regular meetings both virtual and, if possible in person, build a trust-based foundation. During this GSE collaboration, the authors found that various challenges were confidently handled thanks to the mutual trust that has been established over the years. Trust between the authors also made it possible to risk untried solutions.

The latest GSE class was conducted during the winter semester of 2020/21 during the Covid-19 pandemic. Despite the lockdown-imposed limitations, the outcome of this project was on a very high level and a few separate observations and possible implications for future distributed software development can be reported. Further research is necessary to investigate if overall performance can be enhanced by artificially imposing remote work. Although this theoretical framework is not new, this was the first real-world trial. A hybrid architecture should also be investigated, to see the impact of physical teams, virtual teams, and their balance of each other.

Reoccurring pattern could be observed in each GSE class, independent from the course structure. Anticipating the stages in team forming, cultural differences, necessary intervention, and others can strengthen the foundation for successful global software engineering. At the same time, the knowledge of these patterns reduces stress and frustration for the students and instructors.

References

[1] V.K. Agrawal, V.K. Agrawal, A.R. Taylor, and S. Seshadri, "Offshoring it services: Influencing factors," Journal of Management Policy and Practice, vol.20, no.3, Sept. 2019.

[2] G. Hofstede, G.J. Hofstede, and M. Minkov, Cultures and Organizations: Software of the Mind, 3rd Edition, McGraw-Hill, 2010.

[3] E. Hall and M. Hall, Hidden Differences: Doing Business with the Japanese, Anchor Press/Doubleday, 1987.

[4] I. Bosnić, F. Ciccozzi, I. Crnković, I. Čavrak, E.D. Nitto, R. Mirandola, and M. Žagar, "Managing diversity in distributed software development education—a longitudinal case study," ACM Trans. Comput. Educ., vol.19, no.2, Jan. 2019.

[5] T. Clear, S. Beecham, J. Barr, M. Daniels, R. McDermott, M. Oudshoorn, A. Savickaite, and J. Noll, "Challenges and Recommendations for the Design and Conduct of Global Software Engineering Courses: A Systematic Review," Proceedings of the 2015 ITiCSE on Working Group Reports, NY, pp.1–39, ACM, 2015.

[6] J. Carver, L. Jaccheri, S. Morasca, and F. Shull, "Issues in using students in empirical studies in software engineering education," Proceedings. 5th International Workshop on Enterprise Networking and Computing in Healthcare Industry (IEEE Cat. no.03EX717), pp.239–249, 2003.

[7] S. Beecham, T. Clear, J. Barr, M. Daniels, M. Oudshoorn, and J. Noll, "Preparing Tomorrow's Software Engineers for Work in a Global Environment," IEEE Softw., vol.34, no.1, pp.9–12, 2017.

[8] S. Schneider, R. Torkar, and T. Gorschek, "Solutions in global software engineering: A systematic literature review," International Journal of Information Management, vol.33, no.1, pp.119–132, Feb. 2013.

[9] M. Paasivara, K. Blincoe, C. Laasenius, D. Damien, J. Sheoran, F. Harrison, P. Chhabra, A. Yussuf, and V. Isotao, "Learning Global Agile Software Engineering Using Same-Site and Cross-Site Teams," Proceedings of 37th International Conference on Software Engineering (ICSE 15), pp.285–294, IEEE, 2015.

[10] I. Bosnić and I. Čavrak, "Project work division in agile distributed student teams-who develops what?," ACM/IEEE 14th International Conference on Global Software Engineering, 2019.

[11] T. Clear and S. Beecham, "Global software engineering education practice continuum," Special Issue of the ACM Transactions on Computing Education., vol.19, no.2, pp.1–8, Jan. 2019.

[12] R. Hoda, M.A. Babar, Y. Shastri, and H. Yaqoob, "Socio-Cultural Challenges in Global Software Engineering Education," IEEE Trans. Educ., vol.60, no.3, pp.173–182, 2017.

[13] R. Colomo-Palacios, E. Tovar-Caro, Á. García-Crespo, and J.M. Gómez-Berbís, "Identifying technical competences of it professionals: The case of software engineers," International Journal of Human Capital and Information Technology Professionals, vol.1, no.1, pp.31–43, 2010.

[14] D. Joseph, S. Ang, R.H.L. Chang, and S.A. Slaughter, "Practical intelligence in it: Assessing soft skills of it professionals," Commun. ACM, vol.53, no.2, pp.149–154, Feb. 2010.

[15] R. Damaševičius, R. Maskeliūnas, and T. Blažauskas, "Faster pedagogical framework for steam education based on educational robotics," International Journal of Engineering and Technology, vol.7, no.2.28, pp.138–142, SPC, 2018.

[16] T. Ding, "Construction and exploration of university software engineering teaching system based on cdio educational concept," Frontiers in Educational Research, vol.3, no.9, pp.44–48, 2020.

[17] G. Rechistov and A. Plotkin, "Computer engineering educational projects of MIPT-intel laboratory in the context of CDIO," Proceedings of the 10th International CDIO Conference, Universitat Polit'ecnica de Catalunya, pp.1–10, June 16–19 2014.

[18] E.F. Crawley and D.R.B. Malmqvist, William A. Lucas, "The cdio syllabus v2.0 an updated statement of goals for engineering education," Proceedings of the 7th International CDIO Conference, Technical University of Denmark, Copenhagen, pp.47–83, June 20–23 2011.

[19] E.F. Crawley, J. Malmqvist, S. Östlund, D.R. Brodeur, and K. Edström, Rethinking Engineering Education: The CDIO Approach, Springer Nature Switzerland AG, 2020.

[20] D. Jiang and J. Lin, "Project-Based Learning with Step-Up Method —Take CDIO Abilities Cultivation in Computer Specialty for Example," Proceedings of the 8th International CDIO Conference, Queensland University of Technology, pp.1–7, 2012.

[21] A.N. Rodrigues and S.C. dos Santos, "A Framework for Applying Problem-Based Learning to Computing Education," Proceedings of Frontiers in Education Conference (FIE 2016), IEEE, 2016.

[22] J.M. Olivares-Ceja, B.G. Sanchez, P. Brockmann, A. Kress, and J. Staufer, "Project-Based Learning in an International Classroom to Teach Global Software Engineering," Proceedings of International Conference on Education and New Learning Technologies (EDULEARN17), pp.6263–6273, IATED, 2017.

[23] A.-K. Peters, W. Hussain, A. Cajander, T. Clear, and M. Daniels, "Preparing the Global Software Engineer," Proceedings of 10th International Conference on Global Software Engineering (ICGSE 2015), pp.61–70, IEEE, 2015.

[24] R. Vallon, P. Spiesberger, M. Zoffi, C. Zrelski, C. Dräger, and T.

Grechenig, "Teaching global software engineering in a remote customer environment," 2018 IEEE 10th International Conference on Engineering Education (ICEED), pp.63–68, Nov. 2018.

[25] D. Rodríguez, M.A. Sicilia, E. García, and R. Harrison, "Empirical findings on team size and productivity in software development," Journal of Systems and Software, vol.85, no.3, pp.562–570, 2012. Novel approaches in the design and implementation of systems/software architecture.

[26] M. Heričko, A. Živkovič, and I. Rozman, "An approach to optimizing software development team size," Information Processing Letters, vol.108, no.3, pp.101–106, 2008.

[27] B.W. Tuckman, "Developmental sequence in small groups," Psychological Bulletin, vol.63, no.6, pp.384–399, 1965.

Daniel Moritz Marutschke received his M.Eng. (2010) in Human Information Science at Ritsumeikan University and his Ph.D (2014) in Human Communication and Information Science at Kobe University in Japan. He worked as a lecturer (2014–2019) at the College of Information Science and Engineering at Ritsumeikan University and is currently an associate professor (since 2019) at the College of Global Liberal Arts at Ritsumeikan University.

Victor V. Kryssanov received his Ph.D (1994) from the Russian Academy of Sciences and his M.S. (1991) from the Far-Eastern Federal University, Russia. He currently serves as Professor (since 2009) at the College of Information Science and Engineering, Ritsumeikan University, Japan. Before joining Ritsumeikan University in 2004, V. Kryssanov was a JSTA researcher at Kyoto University in 2001–2004, a JSPS research associate at Kobe University in 1998–2001, and a NEDO guest researcher at TRI of JSPMI, Tokyo, in 1996–1998. He also worked at the Far-Eastern Federal University, Russia, in 1991–1995.

Patricia Brockmann received her B.S. degree in Information Systems from the University of Colorado in the U.S.A. She completed her Diplom (M. Sc.) and doctorate in Information Systems at the University of Regensburg in Germany. Since 1998 she has been a full professor in the Computer Science Department at the Nuremberg Institute of Technology in Germany.