ICIC Express Letters
Part B: Applications
Volume **10**, Number **7**, July **2019**

ICIC International ©2019 ISSN 2185-2766
pp. 587–595

JOINT GLOBAL SOFTWARE ENGINEERING EDUCATION IN GERMANY AND JAPAN – A CASE STUDY

Daniel Moritz Marutschke[1], Victor Kryssanov[1]
and Patricia Brockmann[2]

[1]College of Information Science and Engineering
Ritsumeikan University
1-1-1 Nojihigashi, Kusatsu, Shiga 525-8577, Japan
moritz@fc.ritsumei.ac.jp; kvvictor@is.ritsumei.ac.jp

[2]Computer Science Department
Technical University Nuremberg Georg Simon Ohm
Kelerplatz 12, 90489 Nuremberg, Germany
patricia.brockmann@th-nuernberg.de

Received December 2018; accepted March 2019

Abstract. *Engineering education should reflect real-world scenarios as closely as possible. An additional challenge poses global software engineering with unique challenges that need to be faced by distributed teams and tasks. Often due to budgetary constraints and the complexity of conducting global software engineering education, students have limited opportunities to experience genuine and accurate settings in international environments. A global software engineering class conducted by the Ritsumeikan University in Japan and the Technical University Nuremberg in Germany is described with addressing challenges and best practices. A problem-based learning approach provided students experience working in international software development teams. A questionnaire data set is analyzed and interpreted.*
Keywords: Global software engineering, Education, Software design techniques

1. **Introduction.** Global Software Engineering (GSE) poses unique challenges to distributed teams and tasks. In a world that is growing closer in professional environments and interactions, skills to handle such challenges are deemed most important. Due to budgetary constraints and the complexity of conducting global software engineering education, students have limited access to such classes. Studies show that engineering education needs experience and problem-based learning approaches [1-8].

This provides an important experience for future engineers. A difficult task is to keep the learning environment as close to a real-world scenario as possible, in order to expose the students to all of the CDIO stages (Conceive-Design-Implement-Operate) of a modern product lifecycle [6, 7, 9]. The conceive stage is about finding a problem to solve, determining its feasibility and eliciting requirements. Usually students are presented with a problem and guided to feasible concepts. In the design stage, students have to think about the system and program design, i.e., formulating a framework and strategy of how to solve a given problem. The implement stage involves mainly implementing the software, hardware, and any other components to a working prototype. The operate stage is usually addressed only in theory, as it would otherwise exceed the workload. Operation includes maintaining a product over its lifespan, fixing bugs or other problems, reacting to feature requests, adapting to dependent third-party products, and eventually fading out of the market.

DOI: 10.24507/icicelb.10.07.587

The paper summarizes experiences of the latest global software engineering education – taught on a graduate level – at the Ritsumeikan University in Japan and the Technical University Nuremberg and contrasts it to student experiences from undergraduate classes in software engineering.

Following this introductory part, the paper is divided into further three parts. First, in Section 2, the project overview is given, including the educational objectives that guided the course and how the project was organized, namely the class structure, questionnaire and data set description. From Section 3, the main observations are reflected by viewpoints of the two educational institutions – educational professionals and students – with discussions about general and specific difficulties accompanying the project and future propositions.

We provide our insights from questionnaire findings and our personal observations detailing both the Japanese and the German side, and overall discussion respectively. Lastly, we give concluding remarks in Section 4.

State-of-the-art software engineering includes agile information system development, cloud-based and often version-controlled code, documents, and documentation [10, 11]. In distributed collaborative projects, a focus is placed on document sharing and virtual team meetings, usually done by video-chats. Modern IP-based professional video conferencing systems far outperform popular softwares such as Skype, Google Hangout, or FaceTime on video quality, connection stability and often multiple camera angles with speaker tracking capabilities. The gap is, however, closing with professional equipment getting more affordable and readily incorporated with existing video conferencing software.

2. **Project Overview and Methodology.**

This section describes our joint course's objectives and its structure to obtain those goals. Point of views are given by the instructors of both German and Japanese universities, personal insight, and considerations on how to address problems and to improve the course in the future. To support the findings, the authors conducted anonymous questionnaires in all classes to get an insight in the change of students' experience. These questionnaires were specifically designed for classes with problem based learning environments and distributed teams.

2.1. **Educational objectives.**

The main objective of the global software engineering education is to help students to learn the skills necessary for their future roles as engineers in building software with distributed global teams [12, 13]. The skills that stand out and are difficult to acquire with regular didactic methods are listed below.

1) **Conceptual:** Understand key problems in distributed software system development
2) **Tools:** Tools for distributed collaboration, such as cloud platforms, video conferencing software, agile tools, such as Jira
3) **Technical:** Universally understood concepts, such as UML, good programming practices, modern practices of software engineering
4) **Organizational:** Management methods for distributed project groups, distributed agile methods
5) **Intercultural:** Communication with project members from different countries
6) **Ethics:** Team communication, information exchange, respect

The skills above enable the students to model, develop, and document a modest-sized software product while working in an international virtual team. They should be able to apply fundamental computer science concepts and modern IT technologies and tools to designing and implementing software systems. On a more general basis, students should gain the ability to model a software product's properties and make arguments for specific properties using formal and informal logical reasoning. Tacit knowledge is best acquired by practical exercises to appreciate cooperative team skills, ethical behavior, and discuss

contributions and other issues with team members in a professional manner. This includes also the recognition of the cultural diversity of the modern professional environment and the social responsibility of individuals working in a multinational distributed team.

2.2. **Project organization.** This paper gives an insight into a global software engineering class on the master's degree level conducted by the Technical University Nuremberg and the Ritsumeikan University. Two teams were geographically distributed in Germany and Japan with a total number of 14 students. Seven students (German and Columbian) participated at the Technical University Nuremberg, seven students (Chinese, Korean, Vietnamese, South-African) at Ritsumeikan University.

Teams were intentionally designed to be heterogeneous. Each team was made up of half of its members from Japan (who are mostly non-native English speakers from mainly Asian countries), the other half from Germany (all non-native speakers of English). This requires considerably more international communication than if each team is co-located at the same site [14]. On the Japanese side, we let the students form their own groups with two important conditions: 1) friends should not be in the same group and 2) nationalities must be evenly distributed.

Both teams had the chance to meet with a real client who had the need for a customer loyalty system operating an Irish pub in Osaka, Japan. The German as well as the Japanese side had instructors with extensive experience in teaching global software engineering[1] [15-18]. As in most international teams, English was used as a common-ground language.

The one semester structure of the course was as follows. Most classes were introduced by a short lecture in Japan, joined by the German group via Skype. The rest of the time was used for team discussions and feedback with the instructors or with the client. Classes were conducted over 15 weeks (two semester-hours in Japan, four in Germany), with 12 overlapping weeks (joined by both sides). The first half of the semester was dedicated to the design phase, and the second half of the semester was for prototyping.

A voluntary questionnaire was conducted at the end of the semester with a 100% participation rate on both the German and Japanese side. To reflect on the recent questionnaire outcome, questionnaires from previous software engineering classes in Germany – before semester start and after project conclusion – were summarized and taken into consideration. Results and discussions of the questionnaires are described in Section 3.3.

2.3. **Difficulties.** This section lists difficulties with this project, ranging from non-technical ones involving communication skills to network outages and other technical issues.

In the design phase of the project, communication with the client and understanding his wishes and needs proved to be a significant difficulty. Although students were supported and helped, the main obstacle was on the one hand explaining and talking to the client on a non-technical level and on the other hand translating (understanding) the client's propositions into the technical domain. The instructors had to intervene multiple times to point out possible miscommunications and clarify aspects for both the client and student sides.

The system's obtrusiveness on the client side was greatly underestimated. There were too many steps involved, it took too long to verify the customer, distracted the bartender, or had unwanted technical implementations, such as connecting the system with the cash register. During continuous talks throughout the semester, these had to be pointed out to students when the number of steps exceeded the allowed or desired number.

[1]Eight years of undergraduate software engineering courses and four years of advanced topics in global software engineering on the Japanese side, eight years on the German side. The syllabi are accessible via `http://www.ritsumei.ac.jp/acd/ac/kyomu/gaku/onlinesyllabus.htm` and `https://www.th-nuernberg.de/fakultaeten/in/studium/masterstudiengang-wirtschaftsinformatik/` respectively

The implementation phase showed two main problems. One was the teams being stuck in a "status-quo" of technology, i.e., their mindset needed adjustment to explore more possibilities. The other one was the drifting into marketing strategies for the client, rather than focusing on technical aspects of the project. The latter problem reappeared in a different form during the project's final presentation, where an idea to include gamification went too far into business management. Moving into unknown areas can cause multiple issues. Proposed was a feature, which in essence was a drinking game, potentially even opening the bar owner up for liabilities. The instructors pointed this out to students, who quickly recognized the issue and removed it from the final submitted documents. One aspect that was discussed but not followed up on for the final project presentation was security, e.g., a user's malicious intent to crack a system's code or hack the client's database.

Several general difficulties during the 12 weeks, included one network outage (reconnecting after a ten minute break), video-chat quality (partially resolved by sending lecture material previous to each class), and noise pollution while both teams were discussing with their team-counterpart in Germany (resolved by allocating two different rooms).

3. **Results and Discussion.** The reflections on these projects are based on an extensive knowledge of software engineering education and global software engineering education, respectively, on both the German as well as the Japanese side. In the eight years of undergraduate software engineering courses and four years of advanced topics in global software engineering on the Japanese side, and eight years on the German side, collaborations between Russia, Japan, Mongolia, and Mexico have been conducted[2] [15-18].

3.1. **Observations on the Japanese side.** This section details findings and considerations from the viewpoint of the instructors and the students in Japan. The observations reflect on past experiences and feedback unique to the latest project.

3.1.1. *Observations by the educational professionals.* The course was purposefully structured to follow the CDIO stages – Conceive, Design, Implement, Operate. This is to help the students finish with either a working prototype or a concept demonstration. To enforce the real-world implications, one class was dedicated to have either an industry specialist give a speech and lead a Q&A session relevant to the semester topic or, as in this case, to interact directly with a potential client.

The semester is typically divided roughly into two parts, the design phase and the prototyping phase. This is in accordance with the CDIO stages. To focus on the practical parts, individual classes are for the most part structured into one third short lecture and two thirds team discussion with regular interaction with, and feedback from, the instructor(s). The topics that are covered by the lectures are as follows: 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).

Due to university guidelines of how to select courses, a few students drop the class after the first two or three lectures. The instructors experience is that students try to assess

[2]The syllabi are accessible via `http://www.ritsumei.ac.jp/acd/ac/kyomu/gaku/onlinesyllabus.htm` and `https://www.th-nuernberg.de/fakultaeten/in/studium/masterstudiengang-wirtschaftsinformatik/`

the workload of a given class. After reiterating the individual and teamwork efforts, the remaining students have shown to perform very well with a higher than average level of motivation. Their incentive is aligned with the amount of work required for the course.

Reliably, one of the main obstacles is the communication in the class' common language (English). This holds true for both within the teams as well as the virtual teams. This is mentioned to the students several times during the first orientation classes, yet at the end of the semester, two students reported noticeable hindrance related to the language barrier. From experience, it is important to experience different levels of language: 1) people of the same nationality who have different levels of language ability, or 2) people of different nationalities who have different levels of ability. The latter proves to be much more difficult, even with high levels of conversation abilities. This reappeared when five out of six students rated their own English levels as moderate or above.

3.1.2. *Observations by the students.* Feedback regarding the German team members was positive throughout. The main difficulties arose within the Japan-based teams, which might relate to the German students having more lecture time allocated to the global software engineering course. Another strain on the teams posed the heterogeneity of the Japan-based teams, where the German side was mostly homogeneous in terms of common language. Almost all of the German students spoke the same native language, with one exception (Spanish). This is in stark contrast to the Japanese side, which spoke four different native languages, including one native English speaker. The instructors regularly talked to the students related to their team dynamics and encourage mastering heterogeneous team leadership.

The two teams' projects diverged in terms of design and implementation after their first requirements elicitation. Team A had the following problem statement and objectives.

1) **Problem Statement:** The Irish pub does not have a means to facilitate their customer loyalty support program.
2) **Objective:** Design a customer loyalty support solution to fulfill the Irish pub's program requirements.

Team B stated the following.

1) **Irish Pub:** The Irish pub is located in Osaka. Customers are about half Japanese, half foreigners.
2) **Two Branches:** There are two branches in Osaka. Also there might be more branches in the future.
3) **Need of Customer Loyalty System:** The owner needs to build a customer loyalty system, as to encourage customers to come to the pub.

The examples indicated early on that Team A was looking into a future-oriented approach, whereas Team B was following the reliability of proven technology. Team A presented a working prototype programmed in Python based on the Microsoft Azure's platform and used biometrics (face recognition) to identify customers without additional devices or inputs. Team B demonstrated a simple and reliable (and cheap) barcode-based system running on smartphones.

In the early stage the instructors proposed several directions to explore. Although leaving the decision process to the students, a strong interest was in how the project would turn out in a future-oriented system.

3.2. **Observations on the German side.** This section details findings and considerations from the viewpoint of the instructors and the students in Germany. The observations reflect on past experiences and feedback unique to the latest project.

3.2.1. *Observations by the educational professionals.* The main goal of this course at the German university is to teach students the project management and intercultural skills

needed to work successfully on global software projects. Most of the German students had either no experience or only very limited experience working with people from other countries.

A secondary goal was to teach students how to analyze and solve problems on their own. German students are used to having clear directions about exactly how to solve a problem. In industry, however, most customers are not able to clearly articulate their requirements and may not necessarily know exactly what they want until they see a prototype. The German students should also learn how to elicit requirements from customers without an IT background.

An additional learning goal was to learn how to apply software engineering methods in a distributed environment. German students are used to working with agile methods, where all the team members are co-located. The customer is an on-site member of the team and plays the role of the product owner. Here, students had the additional challenge to try to elicit requirements from a customer at a remote location.

3.2.2. *Observations by the students.* Most of the project requirements were presented by the professor at the Japanese university. During one video conference, the customer in Japan answered questions which had been posed by the students of both groups. The German students found the lack of permanent interaction with the end customer to be a disadvantage. German students were used to agile project management, where the product owner is a permanent member of the development team. This limited communication channel made it more difficult for the German students to understand the needs of the customer. The use of the waterfall project management method was also surprising to the German students. Because of the relatively low power distance between students and professors [19], German students expected to ask their own questions and to self-organize their own work.

During the design phase, the students discovered that they had different technological and theoretical backgrounds. As a result, they were accustomed to different design approaches and tools. They found common ground in use-case scenarios, which were used to discuss features of the prototype between the cross-site teams. Internally, each on-site half of each team often used other artifacts to document and propose new ideas for their solutions. The groups also exchanged training videos to teach each other new skills: the German students sent a video on the development environment, and the students from Japan sent a video about the facial recognition system.

A mid-term presentation was prepared by both cross-site teams, but was held by the team members in Japan. Neither the group members nor the professors were particularly satisfied by these preliminary results. The German group proposed to use an agile approach, which could allow for better communication and faster work. Because of geographical and time-zone differences, daily meetings were impossible. A Kanban board and a common group messaging platform for scheduled reports helped to track ideas and enabled multiple students to work on common artifacts.

Each team worked differently. The team members in Japan preferred a clear work distribution method. This meant that only one group member would work on one artifact at a time. The German team members preferred to work on the same artifact together. This approach allowed the teams to manage their own time without losing focus on the work that needed to be completed by the dates agreed upon.

During the review and retrospective at the end of the course, the German students had a chance to discuss and reflect on their experiences. They all greatly appreciated the opportunity to work together with students from another country on a distributed software project. Although the initial challenges in communication and a lack of intercultural understanding proved quite daunting at first, they were quite proud of the fact that they were able to overcome these obstacles to produce their prototypes. The students expressed

their opinion that they learned much more by making their own mistakes than they would have in a conventional lecture.

3.3. **Questionnaire findings.** At the end of the semester, students on both sides were asked to fill out an anonymous survey on their opinions about which factors were most important for global software engineering.

The questionnaire asked for the university degree, nationality, language proficiency, and experience in working abroad or with people of other nationalities. The students were then asked to rank the following factors by their perceived importance: geographic distance, time zone, language difference, proficiency in shared language, cultural difference, familiarity between team members, and trust between team members.

The findings are then contrasted with the same factors from 84 undergraduate global software engineering course students from Germany. Numbers were normalized to fit the one to seven scale of the graduate course.

The results of their perceived importance are presented in Table 1. The questionnaire asked students to rank the factors in order of importance, and both values for each factor are averaged. Smaller values indicate more importance with a minimum of 1, larger values indicating less importance with a maximum of 7. The setup done with Skype as the video-conferencing system reflects trust establishment described in a 2016 paper by Hussain and Blincoe [20]. To underline the importance of mutual respect and trust in the teams and the process, lecturers on both sides spent some time in the beginning of the semester.

Table 1. Comparison of opinions of Japanese and German teams: Most important for global software engineering (*smaller being more important*)

Factor	Japan	Germany	Germany (undergaduate)*
Geographical distance	4.4	5.0	5.46
Time zone	3.3	4.7	4.38
Language difference	3.6	3.9	3.42
Proficiency in shared language	3.1	4.0	3.18
Cultural difference	5.3	2.4	5.21
Familiarity between team members	3.6	5.1	4.23
Trust between team members	2.6	1.8	1.69

*normalized to match scale

Although some stress is inevitable during the project, the instructors talked with the students and pointed out successes that they had by adhering to proven concepts, communication styles and exchange.

The results of this survey show, that students in both Japan and Germany felt that trust between team members was the most important factor in global software development projects. This supports results reported by Hussain and Blincoe [20].

Both teams regarded the geographical distance as one of the lower factors. Although some of the open comments of the questionnaire said they found it frustrating not being able to "just talk in the same room" with their counterparts abroad, cloud systems, video or audio chat, email, and other communication methods make the physical presence less pertinent.

This reflects on today's communication style, where a face-to-face meeting is often replaced by messaging applications.

Time zone difference was average on the Japanese side and lower on the German side. Questions posted by the team residing on the east side would have an easier time to receive on the west side and answer them than the opposite way.

Language differences were rated medium to medium-low on both sides, with a higher discrepancy for proficiency in shared language, where the German side scored almost

one scale-point lower. The biggest variation showed for cultural difference, where the Japanese side put significantly less importance than the German side. This might reflect the heterogeneous group structure within an international course on the Japanese side and the homogeneity on the German side. For familiarity between team members, the Japanese side favored medium, where the German side scored about one and a half scale-points lower.

When put against the findings of undergraduate students, one finding stands out in particular – "cultural difference". In attributing a much lesser value than their fellow, more experienced German students, they rate it on much the same level as the masters level students in Japan.

4. **Conclusions.** From a teaching perspective, there was no significant insight on topics and materials covered. However, feedback from students indicated that team building was one of the most difficult tasks.

A well organized curriculum and problem based education are key in having successful software engineering education. Global software engineering is known to add a few layers of complexity to the mix. Given the natural complexity of multicultural communication, the most difficult task for the instructors is to reconcile individual perception of the team members in the beginning of the project, but also to ensure a fair distribution of project work to team members.

Regular discussions with the instructors proved also to be very important. As in software engineering education in general, there are plenty of possibilities to diverge from a project goal. Another focus of the instructors activity, was to enforce the teams strictly following the course schedule and abiding all the deadlines.

A main focus, which also supports previous research is trust between project members to be identified as the most important factor for the success of global software projects. This was not only the case with students, but trust between the instructors was reported from both sides to be vital for the success of collaborative course in global software engineering.

Acknowledgements. The authors would like to thank Tomas O'Neill for the support of and contributions to the student project.

REFERENCES

[1] J. Barr, M. Daniels, R. McDermott, M. Oudshoorn, A. Savickaite, J. Noll, T. Clear and S. Beecham, Challenges and recommendations for the design and conduct of global software engineering courses: A systematic review, *Proc. of the 2015 ITiCSE on Working Group Reports (ICTICSE-WGR15)*, New York, pp.1-39, 2015.

[2] A. Cajander, T. Clear, A. K. Peters, W. Hussain and M. Daniels, Preparing the global software engineer, *Proc. of the 10th International Conference on Global Software Engineering (ICGSE)*, pp.61-70, 2015.

[3] J. Findlay, A. Weerakoon and N. Dunbar, Integrating multi-disciplinary engineering projects with English on a study-abroad program, *Proc. of the 10th International CDIO Conference*, Barcelona, Spain, 2014.

[4] S. dos Santos and A. Rodriges, A framework for applying problem-based learning to computing education, *Proc. of IEEE Frontiers in Education Conference (FIE)*, 2016.

[5] A. Plotkin and G. Rechistov, Computer engineering educational projects of MIPT-Intel laboratory in the context of CDIO, *Proc. of the 10th International CDIO Conference*, 2014.

[6] J. Lin and D. Jiang, Project-based learning with step-up method – Take CDIO abilities cultivation in computer specialty for example, *Proc. of the 8th International CDIO Conference*, 2012.

[7] R. Sellens, L. Clapham, B. M. Frank and D. S. Strong, Progress with the professional spine: A four-year engineering design and practice sequence, *Proc. of the 8th International CDIO Conference*, Brisbane, 2012.

[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, 2013.

[9] E. F. Crawley, D. R. B. Malmqvist and W. A. Lucas, The CDIO syllabus v2.0: An updated statement of goals for engineering education, *Proc. of the 7th International CDIO Conference*, Copenhagen, pp.47-83, 2011.

[10] M. Hummel, State-of-the-art: A systematic literature review on agile information systems development, *2014 47th Hawaii International Conference on System Sciences (HICSS)*, pp.4712-4721, 2014.

[11] M. Kim, T. Zimmermann, R. DeLine and A. Begel, Data scientists in software teams: State of the art and challenges, *IEEE Trans. Software Engineering*, pp.1-17, 2017.

[12] J. Barr, M. Daniels, M. Oudshoorn, J. Noll, S. Beecham and T. Clear, Preparing tomorrow's software engineers for work in a global environment, *IEEE Software*, vol.34, no.1, pp.9-12, 2017.

[13] Y. Shastri, R. Hoda and M. Babar, Socio-cultural challenges in global software engineering education, *IEEE Trans. Education*, vol.60, no.3, pp.173-182, 2016.

[14] C. Laasenius, D. Damien, J. Sheoran, F. Harrison, P. Chhabra, A. Yussuf, V. Isotao, M. Paasivara and K. Blincoe, Learning global agile software engineering using same-site and cross-site teams, *Proc. of the 37th International Conference on Software Engineering (ICSE)*, pp.285-294, 2015.

[15] A. Kress, J. Staufer, P. Brockmann, J. M. Olivares-Ceja and B. Gutierrez, Project-based learning in an international classroom to teach global software engineering, *Proc. of International Conference on Education and New Learning Technologies (EDULEARN17)*, 2017.

[16] J. M. Olivares-Ceja, M. Harrer and P. Brockmann, Teaching cultural aspects of global software engineering: A virtual Mexican-German team-teaching experience, *Proc. of European Conference on Software Engineering and Education (ECSEE2014)*, 2014.

[17] P. Brockmann, G. Ayurzana, M. Ende and R. Lömmermann, A virtual, global classroom to teach global software engineering: A joint Mongolian-German team-teaching project, *Proc. of International Conference on E-Learning and E-Technologies in Education (ICEEE2013)*, 2013.

[18] P. Brockmann, M. Choinzon, S. Beier and M. Bickel, It takes a global village to teach global software engineering: A joint Mongolian-German team-teaching project, *Proc. of International Conference on E-Learning and E-Technologies in Education (ICEEE2012)*, 2012.

[19] G. J. Hofstede, G. Hofstede and M. Minkov, *Cultures and Organizations: Software of the Mind*, McGraw-Hill, 2010.

[20] W. Hussain and K. Blincoe, Establishing trust and relationships through video conferencing in virtual collaborations: An experience report on a global software engineering course, *Proc. of Inaugural Workshop on Global Software Engineering Education (GSE-Ed'16)*, pp.49-54, 2016.