ICIC Express Letters
Volume **14**, Number **2**, February **2020**

ICIC International ©2020 ISSN 1881-803X
pp. 153–162

DISTRIBUTED TEAMS IN GLOBAL SOFTWARE ENGINEERING EDUCATION: PROJECT-BASED APPROACHES IN BACHELOR AND MASTER DEGREE CLASSES

Daniel Moritz Marutschke[1], Victor Kryssanov[2]
and Patricia Brockmann[3]

[1]College of Global Liberal Arts
Ritsumeikan University
2-150 Iwakura-cho, Ibaraki, Osaka 567-8570, Japan
moritz@fc.ritsumei.ac.jp

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

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

Received August 2019; accepted November 2019

Abstract. *Software engineers in the twenty-first century increasingly need to learn distributed project management and intercultural skills to communicate in international teams. To achieve productive education, software engineering education should reflect real world scenarios as much as possible. In many cases, this proves to be too costly, or neither the infrastructure, nor environment or team culture exists. Recent advances in technology, both hardware and software solutions, however, are facilitating virtual and collaborative teaching and learning environments. In this paper, the authors present a series of cooperative and distributed virtual courses. Implications of new technology and established teaching methodologies are addressed and compared for a bachelor and master degree course in global software engineering. Insights from students questionnaires are gathered and proposed for course improvement.*

Keywords: Software engineering, Smart education, Distributed, Project-based

1. **Introduction.** The difficulties of education revolving global software engineering (GS-E) are well known with a growing field of research. Establishing effective courses is very difficult and makes research and comparisons complicated. The authors have a combined experience of more than 20 years in (global) software engineering, both practical and academic [1, 2, 3, 4]. With a couple of years of collaboration, they formulate findings in undergraduate and graduate level education. Especially in distributed teams, comparative studies lack a wider understanding of conclusions derived from local software engineering teams. Also the subjective evaluation by students and their change thereof after a project is finished are compared. Particularly cultural differences, over-estimation of personal skills, and the importance of trust between teams and team members are comparative to previous findings in software engineering.

To allow for a meaningful education in global software engineering, known factors are implemented in the teaching and learning process. A two-part questionnaire was collaboratively developed to tease out changes or evolution of perception and assessment by students. A detailed view of the questionnaire and its results are described in Section

DOI: 10.24507/icicel.14.02.153

3.2. Another framework that is guiding the teaching close to industry standards is the implementation of CDIO stages (conceive-design-implement-operate). Regardless of local or distributed software development, the mechanisms of a modern product lifecycle have to be understood. Frank et al. analyze a four-year CDIO sequence [5], Crawley et al. investigate an updated curriculum for engineering education [6], and Jiang and Lin write about a CDIO inclusion in project-based learning [7].

A further pillar of global software engineering education is the teaching and learning in the form of projects and problem-solving. Research suggests, that the experience from a practically-oriented syllabus and real-world project-based learning for undergraduate courses and problem-based learning for graduate courses are necessary for effective learning [5, 7, 8, 9, 10, 11, 12, 13]. In many instances, budgetary restraints and the inherent complexity of such projects pose high thresholds. However, with students lacking access to this kind of education, a serious deficit arises. Such challenges are deemed most important to prepare for a world that is growing closer in professional environments and interactions.

The paper summarizes experiences of two recent global software engineering courses – taught on an undergraduate and graduate level – at the Ritsumeikan University in Japan and the Technical University Nuremberg Georg Simon Ohm. The authors provide an insight in global software engineering education practices and how levels in bachelor and graduate courses reflect maturity levels by polling for GSE-relevant questions.

Following this introductory part, the paper is divided into further three parts. In Section 2, teaching goals and objectives are described.

The project overview is given, including the educational objectives that guided the course and how the project was organized. In 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 an overall discussion. Lastly, we give concluding remarks in Section 4.

State of the art software engineering is heavily based on online collaborative tools. Agile information system development, cloud-based and often version-controlled code, documents, and documentation are common place [14, 15]. 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 in terms of video quality, connection stability, and often multiple camera angles with speaker tracking capabilities.

Nonetheless the main reason for failed projects still lies in traditional metrics, such as sloppy requirements elicitation, unprofessional project management, and work-climate, to name a few. Many approaches such as Scrum[1] and Kanban[2] alleviate the most severe failings. However, due to the intricacy of modern software projects, other factors cannot be anticipated without proper training.

2. Teaching Global Software Engineering.

This section details three parts – educational objectives: software engineering in general; in the context of globalization and intercultural skills; with a student-centered teaching method.

The goal of global software engineering education is to teach students the skills necessary to address the difficulties in their future roles as software engineers in distributed teams. Fairly large-scale, systematic reviews of the literature to identify main competences for

[1]Scrum is one of the most used subset of agile project management. Scrum consists of development cycles called "sprints" and is set for a lightweight framework.

[2]Named after the message board system at Toyota's manufacturing, the *kaizen*-inspired continuous improvement strategy was implemented into general project management.

global software engineers, as well as research regarding socio-cultural differences and the need to shift educational goals have been published [8, 16, 17, 18]. Additional skills which students should learn are:

1) Software Engineering Techniques: Understand key problems in distributed software system development, agile development methods, UML, good programming practices;
2) Technical Tools: Distributed collaboration tools such as cloud platforms, video conferencing and project management tools;
3) Organizational: Distributed, agile project management, self-organization in management methods;
4) Conceptual: Critical thinking, logical reasoning to draw conclusions, separating software engineering from tangential fields such as business management;
5) Intercultural: Communication with project members from different countries, reflecting on ones own cultural perspective;
6) Ethics: Respect for all team members, stakeholders, and the environment.

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.

Class structure and project organization. The authors from Germany and Japan conducted a joint master-level course. One author from Japan leads an bachelor-level project based learning (PBL) group between Japan and Sri Lanka. One author from Germany leads a bachelor-level course in introduction to information systems. Due to the data collected, comparisons are drawn between perceptions by the students before and after a project[3]. The following discussions will focus on those countries and mainly referenced unless otherwise necessary.

All teams were geographically distributed with locations in Germany and Japan. The teams were distributed as shown in Table 1. The students in Germany were mostly homogeneous (German, non-native English speakers), whereas teams in Japan were heterogeneous, with students from China, Korea, Vietnam, the United States, Thailand, and other.

Table 1. Number of students in each course

	Germany	Japan
Master-level	9	5
Bachelor-level	19 (15)*	16

*2nd year (3rd year), number of students before and after internship

Teams were intentionally designed to be distributed within teams. Each team was made up of half of its members from Japan, the other half from Germany. This requires considerably more international communication than if each team is co-located at the same site [19]. 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 as evenly distributed as possible. Due to the skew in countries, some groups

[3]In the undergraduate course in Germany, the questionnaires were conducted before and after a semester-long internship.

inevitably had several team members of the same nationality, which led to instances of inner-team incoherence, as reported from the students.

The cross-site project teams in the master-level course were assigned the task to develop a secure e-voting system. The project owner had previous experience with the e-voting system in Switzerland and expert knowledge of e-voting systems in other countries, such as Estonia. Each student group conducted their own requirements engineering, designed and developed a prototype their e-voting system. The course included a guest lecture from a world-leading expert in e-voting and cyber-security. One team conducted initial tests of their prototypes with an eye-tracking system to optimize their website user interface. Another team implemented blockchain processes into their voting procedure.

Distributed teams in the undergraduate level were instructed to create a system that benefits higher education using a wireless camera and image recognition algorithms. Two teams approached this in different ways, conceptualizing a system for automatic attendance logging and an autonomous robot to identify visitors to a designated room.

The one semester arrangement for both master and bachelor level education followed previous structures in global software engineering, with slight modifications to ensure better audio quality and communication. Most classes were introduced by a short lecture in Japan, joined by the German group via Skype. The setup done with Skype as the video-conferencing system reflects trust establishment described in a 2016 paper by Hussain and Blincoe [20]. Slides were shared so students could remotely better follow the lecture. The rest of the time was used for team discussions and feedback with the instructors. The graduate course in Japan was conducted with two semester-hours, the undergraduate course with four semester-hours. Both had a mandatory 15 weeks semester. The graduate course in Germany was conducted with four semester-hours. The undergraduate course in introduction to information systems was conducted during the 2nd and 3rd year (4th and 6th semester).

During the master-level global software engineering course in Germany and Japan, questionnaires were handed out in the beginning and at the end of the semester. These surveys were used to poll students about their changed perspective to several markers related to software development within distributed teams. The same markers were polled in the undergraduate PBL course[4]. A subset of these markers was surveyed before and after a mandatory internship for undergraduate students in Germany[5].

3. **Results and Discussion.**

The results and discussion in this section takes into account the successfully concluded projects, unsuccessful endeavors, difficulties, and lessons learned over a decade worth of global software engineering education. Countries with previous collaborations include Germany, Japan, Russia, Japan, Mongolia, and Mexico[6] [1, 2, 3, 4].

3.1. **Methodology.**

Students on both sides were asked to fill out an anonymous questionnaire. The students were surveyed at the beginning of the semester and at the end of the semester. Questions were polling the students on their opinions about which factors were most important for global software engineering.

The following methodology was used to compare subjective factors in global software engineering:

1) Handing out a questionnaire in the beginning of the semester, polling students on their perceived importance on the following factors: geographic distance, time zone,

[4]The data available for comparison was limited to the one from Japan.

[5]An internship lasting a minimum of 20 weeks up to one semester is common part of the curriculum of engineering education at universities of applied sciences.

[6]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/

language difference, proficiency in shared language, cultural difference, familiarity between team members, and trust between team members.

2) Handing out a questionnaire at the end of the project to poll students on the same factors as above. Additional factors were also surveyed, such as university degree, nationality, language proficiency, and experience in working abroad or with people of other nationalities. A form for open comments was also included where students could address points not covered by the questionnaire.
3) The findings over several years were then evaluated and put into context of global software engineering education. These include individual comments and past research.

3.2. **Analysis and interpretation of questionnaire findings.** Figures 1 to 6 summarize the data as box plots. Evident by the small sample size, many of the attributes are too sparse to make any statistical analysis or draw numerical conclusions. These findings, however, allow for some comparative features and supportive markers for previous research and findings.

The courses were purposefully structured to follow the CDIO stages – Conceive, Design, Implement, Operate. As the syllabus states, this is to "acquaint students with global – social, managerial, and technical – issues that have become increasingly important in modern information and communication technology (ICT) industry." The semester is typically divided 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).

Reoccurring struggles are language barriers and cultural differences. These are usually overcome in the first few weeks of teamwork, with the help and encouragement of the instructors. Conflicts can arise from different viewpoint such as the following:

- the unwillingness to work on weekends (students in Japan were eager to find time to work, even on weekends, whereas students in Germany were keen to keep business within business-days);
- differences in technological infrastructure (students in Japan were surprised when they heard that their team-mates did not have ubiquitous Internet access to have virtual meetings);
- mismatched expectations (1st year students were expecting too much in programming skills from their 2nd year partners overseas).

Perceived and real language barriers are additional factors. In combination of surveying students' own English language abilities and the instructors' experience, perceived language barrier works as follows: English language abilities of an in-group (shared culture and environment) is perceived as higher and having a clearer accent; abilities of out-group members are rated lower with less clear accents.

Surveying the students for eleven factors relevant to global software engineering, Figures 1 and 2 show the distribution of the master-level course at the beginning and at the end of the semester, respectively. Main changes can be noted for geographical distance, cultural differences, communication between teams, and leadership.

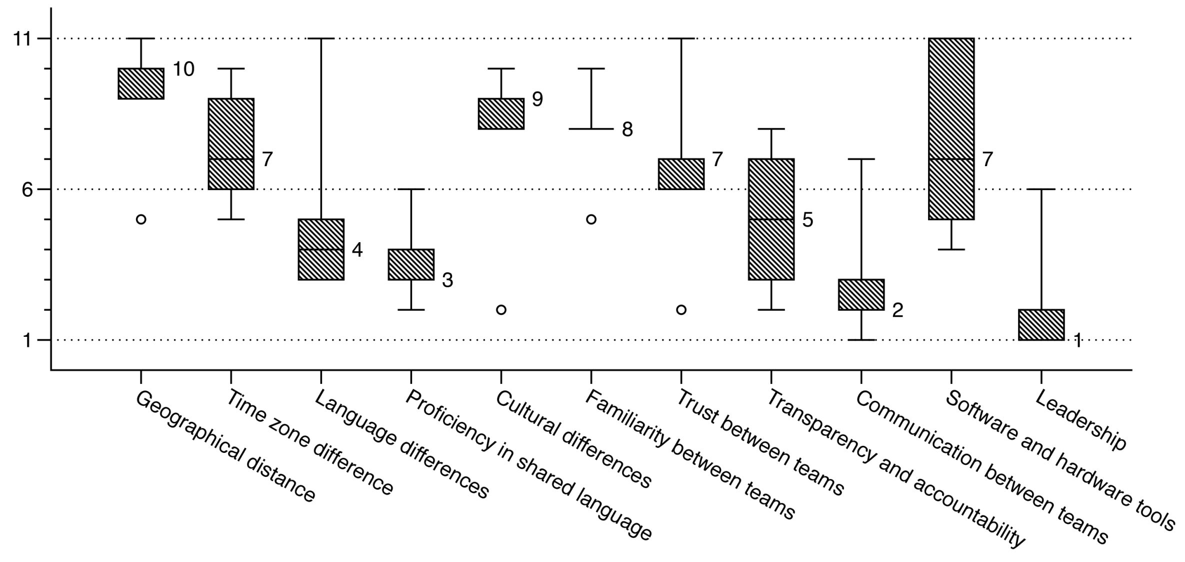

FIGURE 1. Distribution at the beginning of the semester (master-level)

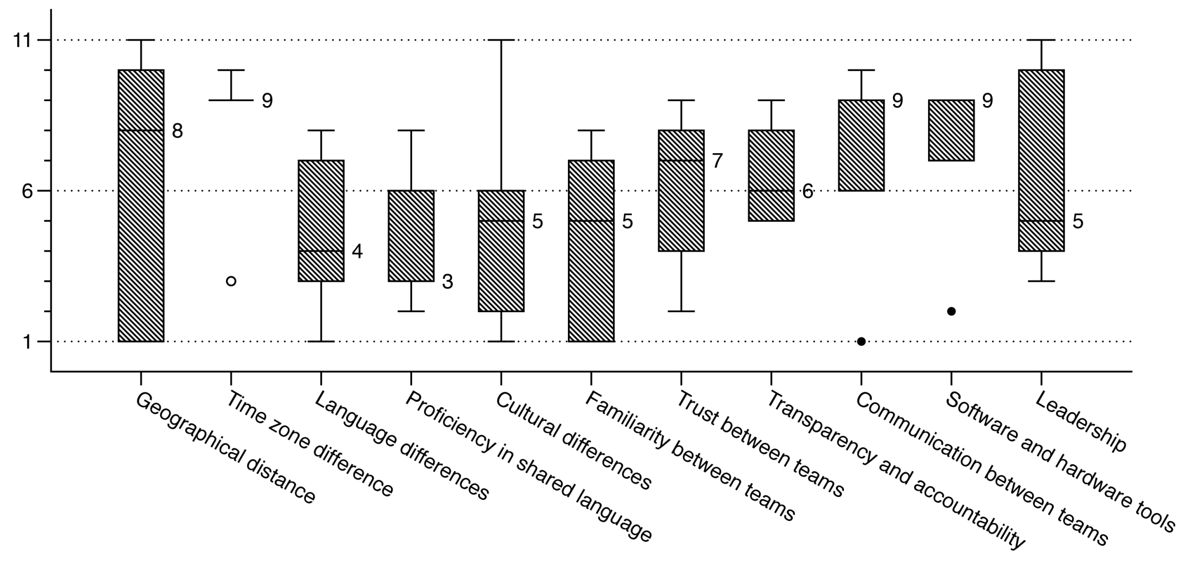

FIGURE 2. Distribution at the end of the semester (master-level)

Once a common project was decided by the teams, this common goal set aside many of the cultural differences or they were integrated into a routine. This includes the geographical distances to some degree.

Interestingly there was a drop in perceived importance (with one outlier rating it most important) of communication between teams. This could tie into the raise in leadership importance, which one team in particular was relying heavily on.

Figures 3 and 4 contrast the bachelor-level students surveyed for a project-based learning course based on global software engineering principals. Most noticeable changes are in geographical distance, time zone difference, and trust between teams. Communication between teams merits an additional mention due to the change in distribution. As the courses at both universities were organized to take place at the same time and with help of cloud-based team management tools, comments and the questionnaire reflect the students' attitude towards differences in location and time zone.

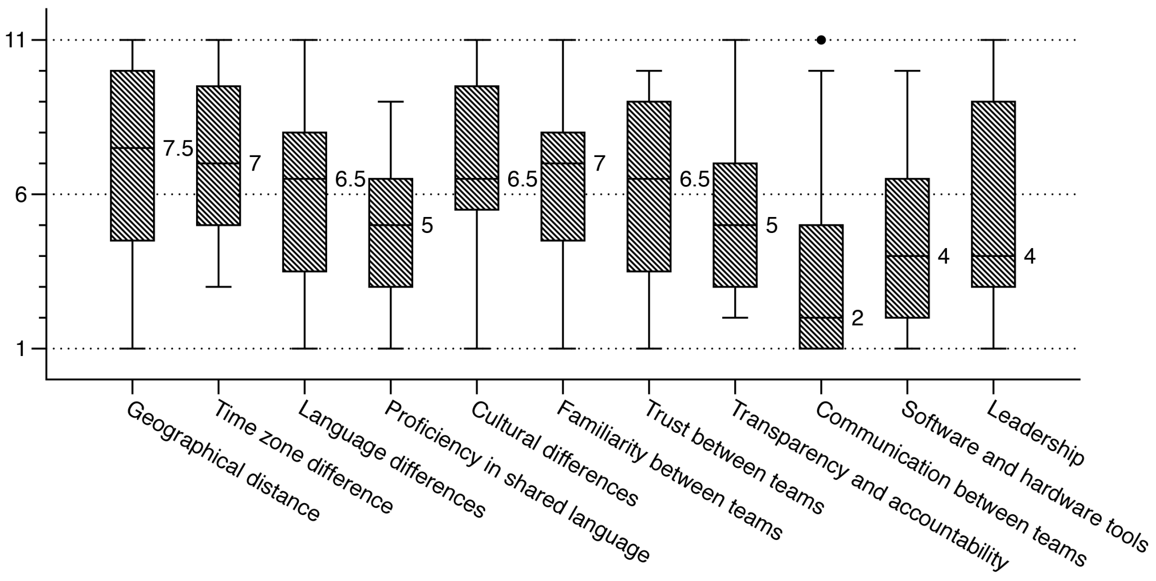

FIGURE 3. Distribution at the beginning of the semester (bachelor-level)

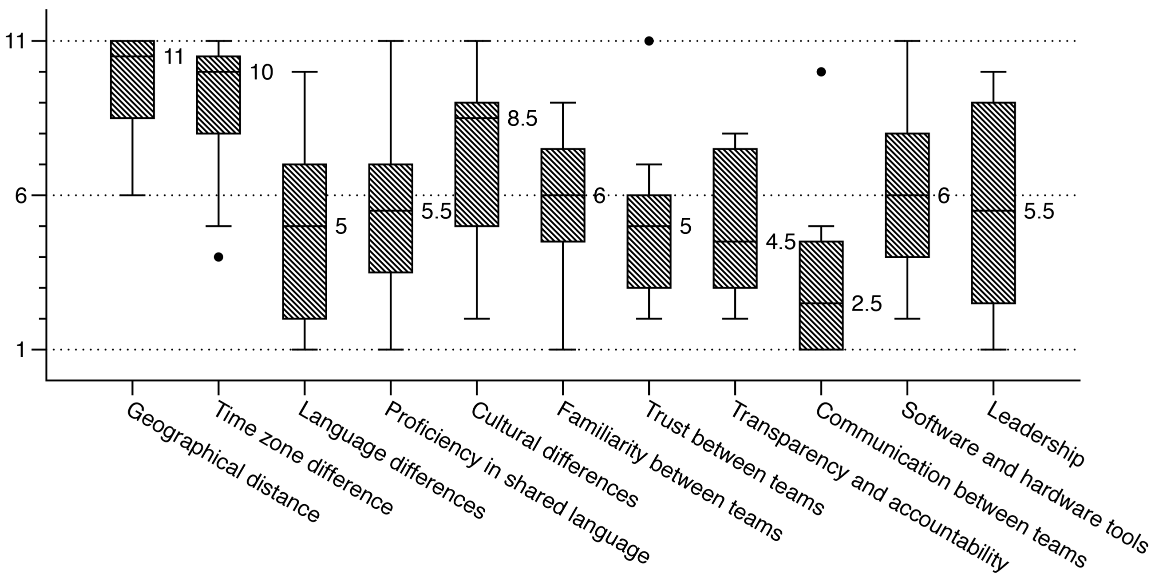

FIGURE 4. Distribution at the end of the semester (bachelor-level)

Trust between teams had an increased perceived importance with the exception of one outlier to rate it least important. Taking comments from students into account, this may be due to unsatisfactory outcome for some team members.

Communication between teams received a cluster around higher importance, with another exception of a single rating least important (10 out of 11).

Overall many of the answers were more concise after finishing the project. The difference to the graduate-level course was also noticeable. Mainly the broad distribution of the bachelor-level students' answers could point to their lack of experience.

An interesting contrasts posed in Figures 5 and 6, which shows surveys of students before and after a mandatory internship and often the first long-term work experience in their study-related industry. The four right-hand factors are only available in the most recent global software engineering projects and are left open for scale and comparability.

The biggest change can be noticed with perceived cultural differences. Being exposed to a real working environment, observed differences can be noticed easier and regarded

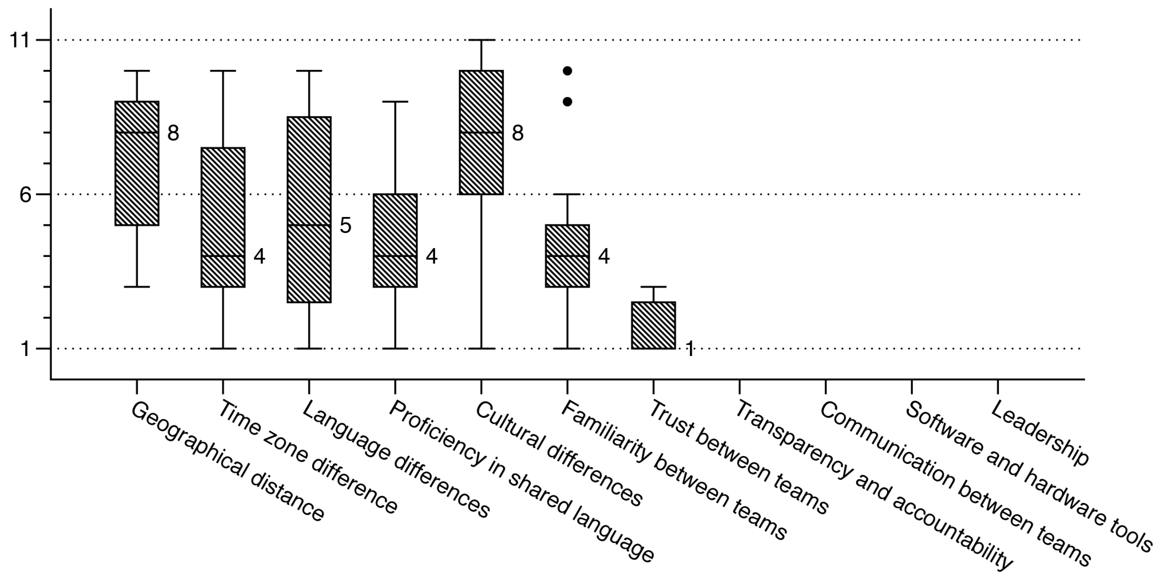

FIGURE 5. Undergraduate course (2nd year) before internship

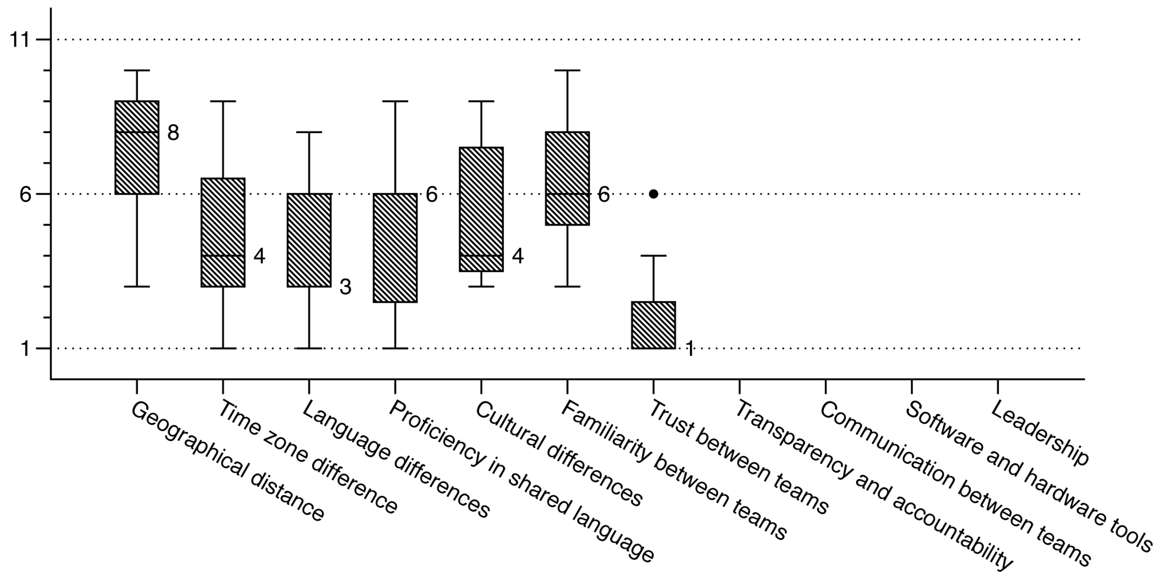

FIGURE 6. Undergraduate course (3rd year) after internship

as potential obstacles. A less pronounced change can be seen in language differences. This might connect to the former experience, but further investigations are needed to form any conclusive remarks. Familiarity between teams underwent a loosening in the accumulation around stronger importance (with previous two outliers).

3.3. **Practical applications.** Global software engineering education is a complex field with a multi-facetted problem space. Understanding previous research and implementing it in one's own experience suggests improving the ability for students to learn the most. A methodological questionnaire can help to identify dynamics in global software engineering and improve education.

4. **Conclusions.** With more observations of students working in distributed teams, the volatility of many perceptions but also reoccurring patterns can be detected. There are consistent changes in attitudes towards cultural differences, time zone differences, and language differences (all becoming less important and therefore less of a reason for

conflict), and a heightened importance in trust between team members. These values and perceptions are difficult to teach in theory and the importance of global software engineering education in an established environment is further supported.

The questionnaire answers within a global software engineering class and around an exposure to a longer-term internship indicate lower emphasis on the former and higher emphasis on the latter experience. This could further suggest the importance of guided project or problem based learning classes in addition to practical skills from a working environment.

REFERENCES

[1] 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*, 2017.

[2] 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*, 2014.

[3] P. Brockmann, G. Ayurzana, M. Ende and R. Lämmermann, 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*, 2013.

[4] 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*, 2012.

[5] 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*, Queensland University of Technology, Brisbane, 2012.

[6] E. F. Crawley, D. R. Brodeur, J. 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*, Technical University of Denmark, Copenhagen, pp.47-83, 2011.

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

[8] 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, *ACM Proc. of the 2015 ITiCSE on Working Group Reports*, New York, pp.1-39, 2015.

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

[10] 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*, Universitat Politècnica de Catalunya, Barcelona, Spain, 2014.

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

[12] 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*, Universitat Politècnica de Catalunya, Barcelona, Spain, 2014.

[13] 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.

[14] M. Hummel, State-of-the-art: A systematic literature review on agile information systems development, *The 47th Hawaii International Conference on System Sciences*, pp.4712-4721, 2013.

[15] 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.

[16] 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.

[17] Y. Shastri, R. Hoda and M. Babar, Socio-cultural challenges in global software engineering education, *IEEE Trans. Education*, no.99, 2016.

[18] R. Maskeliuna, T. Blazauskas and R. Damasevicius, Faster pedagogical framework for steam education based on educational robotics, *Int. J. Eng. Technol.*, vol.7, pp.138-142, 2018.

[19] 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*, vol.2, pp.285-294, 2015.

[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*, pp.49-54, 2016.