

GloSE-Lab: Teaching Global Software Engineering

Constanze Deiters[†], Christoph Herrmann[‡], Roland Hildebrandt[‡], Eric Knauss*, Marco Kuhrmann[§],
Andreas Rausch[†], Bernhard Rumpe[‡], and Kurt Schneider*

* *Software Engineering Group, Leibniz Universität Hannover, Germany*
Email: {eric.knauss,kurt.schneider}@inf.uni-hannover.de

[†] *Department of Informatics – Software Systems Engineering, Clausthal University of Technology, Germany*
Email: {constanze.deiters,andreas.rausch}@tu-clausthal.de

[‡] *Software Engineering, RWTH Aachen University, Germany*
Web: <http://www.se-rwth.de>

[§] *Technische Universität München – Institut für Informatik I4, D-85748 Garching, Germany*
Email: kuhrmann@in.tum.de

Abstract—In practice, more and more software development projects are distributed, ranging from partly distributed teams to global projects with each stakeholder located differently. Teaching actual practice in software engineering at university needs a proper mixture of theory and practice. But setting up practical exercises for global software engineering is hard, because students have to cooperate across different locations and situations reflecting the teaching intentions have to be provoked explicitly.

This paper presents the concepts behind our common teaching environment for global software engineering – the GloSE-Lab. It describes the experiences on setting up a distributed course and reports our teaching intentions based on each universities main focus: project management, requirements engineering & quality assurance, architecture, and implementation. Furthermore, we discuss our setup – a stage-gate process, where each location takes care of a different phase – and report occurred problems and how they supported or interfered with our teaching intentions.

Keywords-global software engineering; teaching; GloSE-Lab

I. INTRODUCTION

Distributed project settings are the typical case in software development today. This results from the fact that in addition to the inherent complexity of software development, there is a huge cost pressure, which increases even further as systems grow. To stay competitive, software development companies make use of price differentials and shift development tasks to low-wage countries. Complex engineering tasks, such as requirements engineering or architecture, typically stay in high-wage countries, e.g., USA or Germany.

Although the understanding of Global Software Development has grown in the recent years, it is not yet a mature discipline [1], [2], [3]. According to [4] 68% of GSD-based teaching/training experiences are presented as case studies, mainly presented by universities. An example for courses/lectures is the Global Studio Project [5]. Accordingly most curricula in Software Engineering (SE) at universities do not concentrate on such distributed project settings. Instead, they usually contain methodological modules and

selected techniques with regards to requirements engineering, architecture, or development techniques. In addition, project management is often covered only in the context of project estimation or planning. Practical courses focus mostly on small groups that work on-site and on synthetic examples. As a consequence, the typical SE curriculum does not respect the changes that occurred in software development throughout the last decade and their impacts on the SE disciplines. To highlight a few examples:

- The management has to respect multi-cultural settings.
- Communication has to be managed across time zones.
- Communication is bound to a language other than the native one.
- Increased complexity of project planning and estimation has to be considered, resulting from an overhead for negotiations and online meetings.
- Architecture needs to respect the system boundaries as well as the geographical and cultural boundaries given by team distribution.

In the fall term of 2010/2011 we coordinated a practical course among our four universities in Aachen, Clausthal, Hanover, and Munich, to provide students with a distributed project setting. The students should learn to work in a distributed environment and make experiences related to asynchronous work, dependencies on other teams without direct access, and communication in a language other than their native one.

Problem Statement: This paper covers the challenges of establishing and coordinating a distributed course. The first of the two core challenges is to create a setting that suits the teaching requirements of each university, while respecting the collaboration with the others. This requires the course to be designed in a fashion that allows for distribution of lecture aspects, since each participating university has its own focus. The corresponding student teams need to fulfill their individual tasks, while the whole distributed project also has a commonly defined goal. The second challenge is

to define a collaboration style that puts the single projects together into an integrated project. Infrastructure is a key aspect to do so.

Contribution: In this paper we describe at first the chosen course setting. We describe how the overall project objective has been defined and how the assignment of responsibilities and tasks was done. Based on the setup we describe our experiences from that course. We outline problems and lessons learned, and derive concrete measures to improve the course format.

Outline: In section II we describe the organizational structure and the technical infrastructure of the GloSE-Lab. We also introduce the system the students have developed. Section III highlights the most relevant problems we observed during the project and also the solutions we applied to them. Furthermore, we discuss our experiences in general and the project as a whole.

II. COURSE SETTING

First, we describe the setting of the distributed course. Table I gives an overview of the participating universities, the number of students at each site, and the tasks the students focused on.

Table I
UNIVERSITIES WITH THEIR NUMBER OF STUDENTS AND TASKS.

Label	Name	S's	Task
RWTH	RWTH Aachen University	13	Development
TUC	Clausthal University of Technology	4	Architecture
LUH	Leibniz Universität Hannover	9	Requirements
TUM	Technische Universität München	3	Proj. Mgmt.

A. Teams & Responsibilities

Each participating university either had a practical SE course in place or created a new one. The single courses correspond to the universities main focus as well as to the SE core disciplines that are required to carry out a software development project. To form the intended distributed course, the single courses were integrated into a distributed setting for our GloSE-Lab. Figure 1 shows the concrete assignment of responsibilities for the considered course. Each course was thereby advised by members of the academic staff. Additionally, each student team chose a team leader through whom most of the communication between the different teams should occur.

B. Technical Infrastructure

To support the student teams of the different universities a technical infrastructure for communication, coordination, and development was provided. This infrastructure was (roughly) determined before the course started.

For communication among the students we provided mailing lists. One list was set up for all team coordinators, one

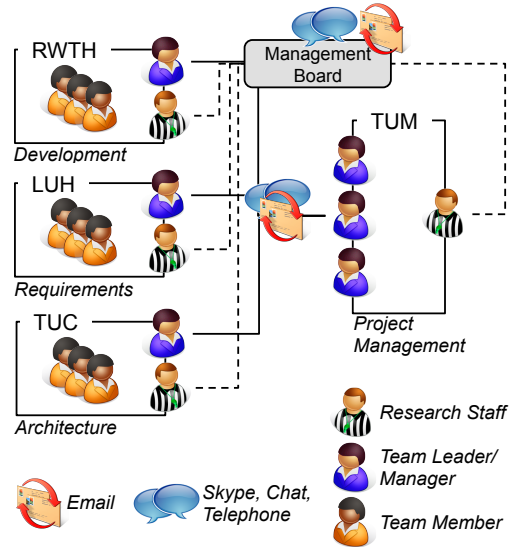


Figure 1. Course setting for the GloSE-Lab: Four teams represent a SE discipline each; a Management Board coordinates the project at a whole.

for all students, and one for the academic staff. Weekly videoconferences were held using Skype or Adobe Connect Pro Meeting. As infrastructure for development a platform called SSELab was used. This is a software engineering platform developed at the RWTH Aachen University, which provides basic project hosting services as well as advanced software engineering services. From the range of tools integrated in SSELab we chose Subversion as version control system, Trac as project-management and bug-tracking tool, and a wiki for documentation purposes. For the implementation task, the students in Aachen were offered virtual developer machines hosted on servers of the university's computing center. This way, the students were provided with a unified development environment with all necessary tools and examples already installed and configured. We did not introduce special tools for global software engineering. The additional training seemed to be too much overhead, especially regarding the limited experience of our students.

C. The Software Development Project

In order to motivate the students, we planned to let them develop a system which most of them are in touch with every day: A social network. This particular network should provide support for distributed SE projects and combine some of the features of Facebook and Twitter. Users of the network should be able to maintain their own profile, e.g., with a picture, their skills, location, or organization they are associated with. Users should be able to add other users as contacts and join project groups to organize the work in their projects. They also should receive status updates that were published by their contacts or within their associated groups.

To have a more realistic setting the requirements engineer-

ing team elicited the requirements for the system from a customer. The customer was represented by a member of the academic staff from the University of Hanover, who was not involved in the course in any other way.

III. PROBLEMS, EXPERIENCES, AND SOLUTIONS

The goal of the GloSE-Lab is to let students experience (and solve) real world problems in distributed software projects. Therefore, we wanted some challenges to occur. Of course, there were other problems that did not support the teaching experience. A distributed software engineering course has to face the risk that too many problems make the course a frustrating experience with only marginal learning effects. In this section, we share our experiences and discuss strategies our students applied to solve their challenges. Afterwards, we discuss the challenges from a teaching point of view. We show how we managed the risks, which unintentional challenges occurred, and give suggestions how similar courses can avoid these pitfalls.

A. Basic Facts

As Table I shows, 29 students participated in the GloSE-Lab. The lab was scheduled all in all for 16 weeks. The lecture period of each involved university started at different dates, which proved to be a major challenge to our schedule. During this time, 10 weekly video conferences were carried out by all universities, and around 100 emails were sent via the global mailing lists. Interestingly, the students spend more effort for internal communication (ca. 400 emails and one hour communication via Skype in Aachen, ca. 300 emails in Hanover, and ca. 150 emails in munich). This was caused by the fact that the local teams also worked in distributed sub-groups.

While the issue management system Trac was offered to both the global project management and the team managers in Aachen, the team managers in Aachen employed it in a greater extend. Overall they issued 109 Trac tickets during the implementation phase of their course. In comparison, the global Trac contained 24 tickets for all teams together.

B. Project Progression

Figure 2 displays a trend analysis of the GloSE-Lab milestones. For each milestone of the stage-gate process – i.e., requirements, architecture, first prototype, and final delivery – the expected date (y-axis) is recorded at each reporting date (x-axis). Some interesting facts/effects we observed are also visible in this figure:

- The “waterfallish” stage-gate approach is visible by the initial set of milestone dates (i.e. the y-axis).
- In the beginning, the students rescheduled several times. E.g., the delivery of the requirement specification was delayed in small dozes several times in a row.
- After the first adjustment in November, the architecture team did not reschedule their delivery until shortly

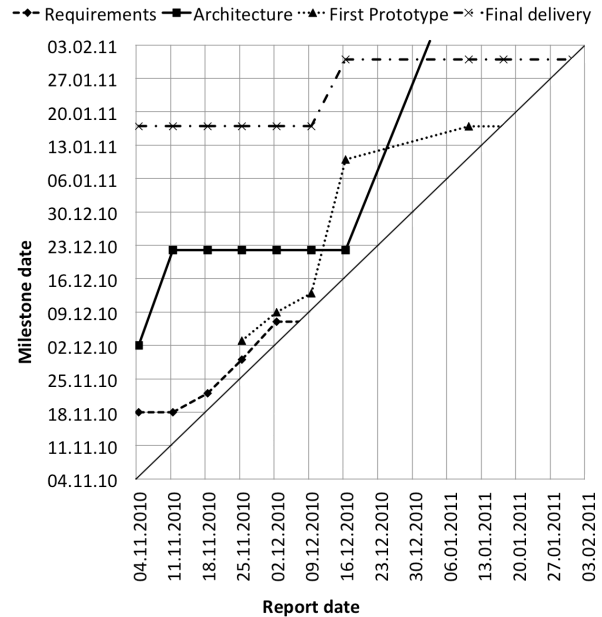


Figure 2. Trend analysis of the GloSE-Lab milestones. No updates where recorded during the Christmas break (23.12.2010–06.01.2011).

before the milestone was due, but then they were detached from the overall project.

- The milestone “first prototype” was not planned at the very beginning of the project, but was introduced when the architecture was first delayed.
- Noticeable shifts occurred in the middle of the project (middle of December):
 - At this point in time the architecture team started to be detached from the main development process.
 - The first prototype was decided to be used as basis for the final delivery.
 - The final delivery was rescheduled due to the delay of the prior milestones. As the students had learned to estimate what they could accomplish in a certain period of time, they also decided to disregard some of the specified requirements.
- The product was actually delivered at the date that was set in December, but – as mentioned – it did not realize all the specified requirements.

C. Observations, Reasons and Workarounds

In addition to the problems mentioned in section III-B, we outline more of the observations we made during the GloSE-Lab. Furthermore, we identify reasons for these problems and present strategies that we as well as our students developed to compensate some of these shortcomings.

1) *Broken Schedule:* The four universities had different start and end dates of their lecture periods, so that the available time frame for the course was already narrowed

compared to a traditional non-distributed course. In the beginning, the students underestimated the necessary effort – due to missing experiences – and suffered from a “next week syndrome”. I.e., they promised to finish a certain task the next week but postponed it again when the delivery was due. One reason for this was an implicit time table without strict delivery dates. Another reason was the absence of consequences for teams that missed deadlines. This can be attributed to the fact that it was unclear who was responsible to impose sanctions in such a situation (see also *Unclear Roles*). Because of the postponed deliverables the teams that depended on them became impatient. E.g., the implementation team increased the pressure on the architecture team. As a consequence, this made the latter feel insecure and did not lead to further progress. In the end the implementation team gave up waiting and started without the deliverable of the architecture team.

2) *Communication Challenges*: The teams were challenged by organizing themselves and communicating internally with each other, and also by communicating with the other teams. E.g., they underestimated the communication overhead and delay. Additionally, the inter-team communication was further complicated by choosing English as working language. This choice had to be made as not all students could speak German. This fact led to reduced communication between the teams, e.g., especially between the architecture and the implementation team. To mitigate the communication situation, the project management team tried to mediate between the two teams. Overall they did not succeed, since the progress of a single team was not clearly visible to the others due to insufficient usage of the provided communication channels and management tools, e.g., the Trac system. Closely related to such communication issues is also the *Technology Question* problem.

3) *Technology Question*: Due to a lack of further information the requirements team assumed that the customer would decide which implementation technology to use (the Play!-Framework). This led to conflicts with the implementation team which insisted on the technology they were already trained in (Java EE). This problem arose because the project manifesto was written before the start of the GloSE-Lab. Only the latest version clearly stated which technology (Java EE) should be used. Unfortunately, the ownership of this document was unclear and nobody took responsibility to ensure that the most recent version was used by all students groups. Thus, this version was not available to the requirements team.

4) *Indefinite Project Resources*: Whenever the requirements team asked the customer to validate the specification, he added new requirements. Therefore, the number of requirements and derived use cases reached an amount which could only be implemented partially. The requirements team was unable to solve this problem they did not perceive effort estimation as their job. The other teams did not feel

responsible, too. Thus, the distributed team as a whole was unable to limit the scope of the project. In the end, the requirements team provided a list with priorities for the different functional requirements and the customer was made aware of the limited resources of the project by the academic staff. In the context of this conflict, another organizational communication problem surfaced. Because of unclear escalation paths the requirements team first contacted the project management team and not their advisers directly. Contacting the responsible persons and solving this conflict led to a delay. Finally, both the architecture team and the implementation team made a compromise between two options: partly implementing as many use cases as possible or implementing only a few use cases in their entirety.

5) *Unclear Management Roles*: Unclear defined management roles and responsibilities led to some confusion. E.g., it was unclear who was responsible for setting deadlines and who had to enforce these deadlines. Some participants tacitly thought this task would be assigned to the project management team. But unfortunately, this issue was not discussed during the GloSE-Lab, but showed an important issue: Establishing “real” project management requires the ability to give orders and to initiate sanctions. We are currently unaware, how to setup a student project with that mass of “power”.

6) *Knowledge Deficit*: Practical courses like the GloSE-Lab are substantial for the education of students. Often such courses represent their biggest projects so far for most of the students. Hence, the teams did not have much prior experience within their domain. Because of the phase-based approach, the implementation team could use the timeslot before the beginning of the implementation task to get trained in Java EE. The architecture team did not have such a timeslot they could specifically use for training, and additionally they suffered from the least prior experiences. Therefore, necessary training during the beginning of the GloSE-Lab delayed their start of the actual work and the delivery of results. Finally, they were detached from the project (see *Lost Team*).

7) *Lost Team*: Around the end of December the architecture team was left behind. There were different reasons for this. Beside the language barrier their knowledge deficit was considerable and they were not able to compensate this in a short period of time. Their detachment from the other teams frustrated not only themselves but also the other teams, like the project management, which vainly tried to reach them. To finish the project with results, the implementation team extended their prototypes to final deliverables without an architecture specification. Meanwhile, the architecture team worked out an architecture specification which was not included into the cooperative work.

8) *Unequal Workload*: Over the time the workload of different teams was unequally distributed. The requirements team started with the requirement specification. But until

they were finished, the other teams could not start to work. And afterwards, the requirements team had no more tasks. To mitigate this effect, we introduced prototyping phases for the requirements team and the implementation team.

D. The students' point of view

Due to different motivations and perspective, the point of view of the participating students might differ radically from our point of view as advisers (as described before). Therefore, we also asked the students to share their observations and opinions. This information was obtained by questionnaires, from intermediate and final presentations given by the different students groups, and during project debriefing sessions. Altogether, their answers match our observations and opinions. Ranging from scheduling issues (“Set realistic deadlines that must be held by the teams.”, “Give to some teams some ahead time.”, “Illness and unforeseen difficulties led to delays because no time buffer existed.”) to the amount of specified requirements (“Too many requirements.”) they also addressed communication issues with partially different opinions (“Drawbacks: Video conferences were only for status updates; Email correspondence did not include all interested parties; Misunderstanding and misinterpretations”, “[I] think that the weekly conference calls were reasonable and very good.”, “Due to English as project language solving questions took more time.”). Also proposals for further projects of this kind were made, like “The project management should work closer with the supervisors.”. Beyond that some comments regarding their individual experiences are mentionable: “[It was positive to] practice negotiation and agreement settlement” and “I felt really proud to say to my colleges that I was going to have a teleconference that day with other Universities (I mean every Thursday) [...]: it's impressive, isn't it?”.

E. Discussion

A distributed course raises more and partly different problems than an undistributed course. Beside their core tasks the students were also challenged by a lot of problems referring to communication issues and organizational coordination. Thus, they could experience how social aspects affect communication. A foreign language as project language can be a chance to extend their language skills, but it can also hinder communication and threaten the project success.

Furthermore, unforeseen problems forced the students to adjust their work and schedule. Already during the first weeks of the course the students gained experiences and made necessary adjustments. They could work out the adjustments on their own and could observe the impact of their decisions in the following weeks.

The success of such a course stands or falls by the knowledge level of the students (see: *Lost Team*). To avoid this drawback, concrete prerequisites (e.g., project management, language, technologies and methodologies) should be

defined for future courses and if necessary, local training should be performed before the global kick-off.

Due to different locations some basic facts could vary and need to be considered adequately. E.g., our universities have different start and end dates of the lecture periods, which we did not realize until the project already started. This point also uncovers the need for an improvement of the communication between the academic staff.

Lessons Learned: Summing up our experiences we strongly suggest to keep the following issues in mind when executing similar distributed courses:

- *Clear Schedule.*
- *Choice of Technology.*
- *Clear roles, responsibilities, and escalation paths.*
- *Knowledge prerequisites.*

IV. CONCLUSION

In this paper we presented our experiences of a distributed practical course that displayed typical problems of global software engineering and also approaches for overcoming them. Motivated by the positive and negative feedback of all participants, we are planning on additional GloSE-Labs. For the future, we plan to further extend the course format and invite other universities to the GloSE-Lab. Based on our experiences we are now ready to invite universities from other countries or even from other time zones to create a setting that matches much more the industrial reality. For today, we hope that others will already benefit from our experiences and improvements we suggest. Most of these suggestions relate to communication and coordination between the students, but also between the academic staff. In the end, the organization of a course for global software engineering is also a distributed project.

REFERENCES

- [1] D. Damian and D. Moitra, “Guest editors’ introduction: Global software development: How far have we come?” *IEEE Software*, vol. 23, 2006.
- [2] J. D. Herbsleb, “Global software engineering: The future of socio-technical coordination,” in *2007 Future of Software Engineering*, 2007.
- [3] C. Bartelt, M. Broy, C. Herrmann, E. Knauss, M. Kuhrmann, A. Rausch, B. Rumpe, and K. Schneider, “Orchestration of Global Software Engineering Projects,” in *Proceedings of the Third International Workshop on Tool Support and Requirements Management in Distributed Projects (REMIDI’09)*, 2009.
- [4] M. J. Monasor, A. Vincaino, M. Piattini, and I. Caballero, “Preparing students and engineers for Global Software Development: A Systematic Review,” in *International Conference on Global Software Engineering (ICGSE)*, 2011.
- [5] I. Richardson, A. E. Milewski, P. Keil, and N. Mullick, “Distributed Development – an Education Perspective on the Global Studio Project,” in *28th International Conference on Software Engineering (ICSE)*, 2006.