T. Koschmann, D. Suthers, & T.W. Chan (Eds.), *Computer Supported Collaborative Learning 2005: The Next 10 Years!*
Mahwah, NJ: Lawrence Erlbaum Associates.

Teaching Distributed Software Development with the Project Method

Till Schümmer, Stephan Lukosch, Joerg M. Haake

Computer Science Department
FernUniversität in Hagen, Germany

Corresponding author: Till.Schuemmer@fernuni-hagen.de

Abstract. Lab courses are an integral part of higher education in engineering sciences. In this paper, we report about a blended learning approach for such courses in computer science education. We show how the project method for co-located learning can be translated into a blended CSCL setting for a lab course on distributed software development, and how the resulting learning scenario can be supported with the collaborative virtual learning environment CURE. Experiences from the application of the educational approach in two courses indicate benefits but also highlight the need for further technical support.

Keywords: CSCL, project method, blended learning, distributed lab courses, collaborative virtual learning environments, CURE.

INTRODUCTION

In the context of computer science, lab courses on programming are an important part of most degree programs at universities. They foster understanding of programming concepts by actively applying theory in the context of a problem that is comparable to a real-world problem. By situating the learning objectives in a real-world context, the experience becomes more authentic and easier to reuse in later industrial settings. Thus, the learning shifts from knowing *what* has to be done to experiencing *how* problems have to be addressed.

The *Problem-Based Learning* approach (PBL) (Woods, 1994) provides an educational background: Students are encouraged to collaboratively find a solution for a problem. The problem is created by the instructional designer and should be close to reality. PBL has gained major interest especially in engineering faculties (Hadim & Esche, 2002). Addressing practical education in software development with a PBL approach naturally leads to the following structure for collaborative programming labs, which can be often found in practice:

- The teacher announces the lab course together with a description of the task.
- Students register for the lab course.
- Student groups form and norm (Tuckman, 1965).
- Students collaborate following a prescribed problem solving process (i.e. a specific design method such as the Unified Software Development Process (Jacobson, Booch, & Rumbaugh, 1999))
- Students submit deliverables at prescribed milestones.
- The lab course finishes when students have delivered the final milestone and received a grade.

German examples for lab courses run in this way are the eXtreme Programming course at the University of Karlsruhe (Bunse, Feldmann, & Dörr, 2004), the software engineering lab at the Technical University of Darmstadt (Schroeder, Brunner, & Deneke, 1998), or the lab course on databases at the FernUniversität in Hagen (Becking et al., 2004). Similar courses can be found in other countries. In the first two cases, the lab course takes place at a traditional campus-based university. Student groups are able to meet in person and frequently do so. Practices like design meetings or pair programming sessions (Williams & Upchurch, 2001) frequently take place at the University or at student's homes (in the example of Darmstadt). Students have to follow a design method that was described in a lecture before the course. In the the eXtreme Programming lab (in Karlsruhe), tutors closely interact in the process to ensure that the students stick to the roles prescribed by the eXtreme Programming methodology (Beck, 1999). The tutor involvement during pair programming and planning implies that most collaboration takes place at the same place (i.e. a lab at the campus).

An example for a comparable course in a distributed setting is the databases lab. The group process is similar to what is used in the co-located setting, but students interact in a virtual environment (BSCW (Appelt & Mambrey, 1999) and IRC). It is not mandatory that students meet in person at any time during the course.

Again, the process is prescribed by the teachers including information about the required deliverables. All teams have to solve the same task (e.g., design a web-based information system for the university) but have the freedom to distribute work among team members according to group needs. Teachers act as customers and observe the team process. They only intervene if the group is in danger of applying the process incorrectly.

From our experiences at the FernUniversität in Hagen, we see several problems that make the application of the above model difficult in distance teaching programs (such as those offered by the FernUniversität), where students are distributed throughout Germany and other countries, and in most cases have a full-time job that allows only limited time for engaging in the lab course. The problems are:

- motivational problems, e.g. where the students' jobs are more relevant than the lab course,
- schedule problems, e.g. where different courses make it hard to follow yet another prescribed schedule,
- process problems, where traditional processes like the Unified Process are considered as too heavy-weighted for the relatively small task in the lab, and
- communication and interaction problems, where the groups experience difficulties in the group interaction because of deficits in the group building (formation) phase. Weak group building makes it more difficult for group members to understand their peers. This leads us to the opinion that social practices should be learned during a distributed software development lab. These include mastering of distributed teamwork, which we consider as a key qualification in a global environment.

All these problems lead to a high drop-out rate. Due to the fact that the course is based on group work, the drop-out is even more critical since the remaining group members have to restructure their work and often cope with a higher workload after one member left the group.

In this paper, we report on an alternative educational method for distributed labs in the context of software engineering that is based on the project method. We will first provide an overview of the project method and then present our proposal for the application of this method in the context of software engineering education. Then, we show how we implemented the method using the CURE environment (Haake et al., 2004). Finally, we report on our experiences from two instantiations of the course, discuss the experiences with respect to related work, and provide directions for future research.

THE PROJECT METHOD FOR DISTRIBUTED SOFTWARE ENGINEERING

As Knoll (1997) pointed out, the *Project Method* as it is now widely used in Europe has its roots in architecture education in the late 16th century. Nevertheless, the first influential essay on the project method was published by the American educational theorist Kilpatrick (1918) who redefined the term project as a *purposeful act* in the context of child education. The main objective was to allow children to acquire knowledge in practical and social contexts. Though Kilpatrick refers to children, the same applies to the education of adults. According to Kilpatrick, projects should be run in four phases:

- **purposing**, where the student brings in a project idea/goal,
- **planning**, where the the required steps for solving the proposed problem instance are identified,
- **executing**, where the steps are performed, and finally
- **judging**, where the mature student judges the outcome of the process and compares it with his initial project goals.

It is important that all phases should be executed by the students, not the teacher. The teacher provides help where necessary, but the students design the goals and procedures of their process. This is the main difference to the common educational setting outlined in the introduction.

Gudjons (1997) identified eight important characteristics for applying the project method. These characteristics have been refined by many educators and even made their way into recommendations of the Austrian ministry for education (Kölbl, 2001). A project should be

- focusing on the interests of the main participants, namely the students *and* the teacher,
- self-organized by the students in agreement with the teacher,
- following a goal-oriented planning process in the group,
- interdisciplinary with respect to the learning objectives,
- a highly social process that matures social competences of all participants,
- with external effect to the student's environment (which means that the result should be of importance for the participants),
- using cooperative rather than instructional joint activities between teachers and students, and finally,
- activating different senses.

Although these characteristics can be found in isolation also in other educational methods, the project method combines them in a way that is very relevant for collaborative learning. On a theoretical level, the different characteristics of the project method help to overcome the obstacles mentioned in the introduction. Motivational problems are addressed by the purposing phase of the project, which mainly focuses on self defined tasks, goal orientation and the external effect. Schedule problems are tackled by the planning phase of

the project method. Since the planning is in the hands of the students, it can be modeled as an iterative activity, meaning the plan can be constantly rescheduled with respect to the group's needs. Process adequacy is approached both in the planning and the execution phase. Students are encouraged to constantly reflect and reshape their work processes and thus learn to improve the process. For reflection, social skills are as necessary as content skills and therefore teachers provide comments on the current process as well as help for students having problems to fill their roles.

For using the project method in a distributed software development lab, we propose the following sequence of actions, which are performed in distributed or co-located settings (as indicated):

Administration (distributed)	
Announcement	Teachers announce the course in the course directory. A supplementary description contains more details on the problem space that can be addressed by the student projects.
Enrollment	Students decide to participate in the course and register for the course providing more details about their background knowledge.
Selection	Teachers select those students that fulfill the knowledge requirements.
Purposing (distributed)	
Project idea generation	Teachers provide the students with a set of requirements that have to be addressed by all proposed projects. These requirements ensure that each project contains all aspects that are part of the learning goals. Students create project outlines for projects that they like to work on. To achieve a larger variety of ideas duplicate ideas are discouraged.
Project idea discussion	Students comment on the ideas generated by other students. The discussion should clarify the individual ideas and provide a shared understanding of possible projects.
Project idea screening	Teachers select the most promising project ideas (since the project should be of relevance for the teacher and for the students, it is important that only ideas are considered for presentation that are in line with the course's goals).
Presentation preparation	The students who created the selected project ideas prepare a short talk advertising their project outline for the first co-located phase.
Group building and Planning (co-located)	
Socializing	An important issue for students who normally do not meet in person is socializing. The students need to get to know one another before they can decide with whom they form a group. Thus, students exchange information concerning personal skills and project preferences. Examples for personal skills are project management or network programming experiences. A good means for socializing can be games although they need to match the social structure of the group (McKee, Solas, & Tillmann, 1998).
Project idea presentation	The students present their ideas in a co-located plenary meeting. After each presentation, there is time for clarifying questions.
Group formation	Students form groups of 5 to 7 people with regard to the information they have of each other concerning personal skills and project preferences. Examples for personal skills are project management or network programming experiences. Students are asked to consider not only the preferences for specific project ideas but also look for a well-balanced distribution of skills within the group. A well-balanced group should at least include one member interested in carrying out the project management.
Project selection	The groups discuss different options for projects and create proposals for their most prominent three projects. The proposals should already include an outline of the intended approach to the project. Teachers then assign projects to groups based on the quality and credibility of the proposals.
Introduction of methodology	Teachers present selected software engineering methodologies. They are intended as an input for the students who are then asked to create their own methodology matching the problem. The methodology addresses problem-specific issues (like programming styles) and social issues (like communication guidelines).
Creation of work plan	The problem is decomposed into small work packages. Students are asked to make first estimates on the required efforts for each work unit. Work units are then put into a workflow.
Assignment of roles	Students assign themselves to roles (identified in the customized group process), areas of expertise, and work units. Each role and area of expertise should be covered by two students to encourage knowledge transfer in the group and to reduce the consequences of drop-outs.

Table 1 (part 1): Phases of the project method.

Execution (distributed)	
Group work	Group members work on small work packages in small subgroups or individually (depending on the possible means for distributed collaboration and the complexity of the task). The students' results are shared and discussed frequently.
Monitoring	Teachers and the project leader monitor the work plan and ensure that the group is aware of any delays in the plan. Group members are asked to recalculate their estimates if the delay increases steadily. The project leader is in charge of monitoring group members' behavior so that they stick to their roles. This does not mean that they may not assist other group members in other roles, but they have to ensure that the responsibilities bound to their role are respected. Teachers only intervene if the project leader does not stick to her role or if the project leader asks the teacher for help. If the group recognizes that specific roles are not well assigned, they can decide to switch the roles (but not frequently).
Inter-group exchange	The groups are also asked to exchange insights regularly. Especially, when different groups have to create software based on the same technology and facing similar issues, this exchange can lead to mutual learning between the groups.
Judgement (co-located)	
Presentation of results	Each group prepares a product presentation. The other groups and the teachers provide ideas for improvement for the groups.
Reflection	Students compare their initial goals with the current state of the project in a project retrospective (Kerth, 2001). They are asked to report on parts of the project work that worked well and parts that failed, e.g. communication or project management issues. The role of the teacher is to point the group to good or bad aspects in their process if the group does not find these aspects on their own.
Grading	Together with the group, the teacher awards a grade based on the insights that students gained.

Table 1 (part 2): Phases of the project method.

SUPPORTING BLENDED LEARNING IN A PRACTICAL LAB COURSE IN CURE

We used the CURE (Collaborative Universal Remote Education) collaborative learning platform to support the above method in distributed software development lab courses. In the next sub section, we introduce the main concepts of CURE. Then, we present how the project method described above is supported in CURE.

CURE in a Nutshell

CURE is based on the metaphor of virtual rooms. Room metaphors (Greenberg & Roseman, 2002; Pfister et al., 1998) have been widely used to structure collaboration. The room metaphor uses the room as the representation of a virtual place for collaboration. Rooms can contain pages (content), communication channels (such as chat, threaded mailbox), and users (see figure 1). Users, who are in the same room at the same time, can communicate by means of a synchronous communication channel (i.e. a chat) that is automatically established between all users in the room. They can also access all pages that are contained in the room. Changes of these pages are visible to all members in the room. The concept of a virtual key is used to express access permissions of the key holder on rooms (such as the rights to enter a room, create sub rooms, edit pages, or to communicate within the room). Rooms with public keys are accessible by all registered users of the system.

Users can enter a room, whereby they can now access the room's communication channels and may participate in collaborative activities. Users can also create and edit pages in the room. Pages may either be directly edited using a simple WIKI-like syntax (Leuf & Cunningham, 2001), or they may contain binary documents or artifacts. In particular, the syntax supports links to other pages, other rooms, external URLs or mail addresses. The server stores all artifacts to support collaborative access. When users leave the room, the content stays there to allow users to come back later and continue their work on the room's pages.

Figure 1 shows a typical room in CURE (the numbers refer to details explained in the following section). It contains documents (in the example, the user Frank reads the document "Homepage" in the room "GoGoGadgeto" – 1) that can be edited by those users, who have sufficient edit rights (2). It provides two room-based communication channels: a mail box (3) and a chat (4). Users can use the room-based e-mail to send a mail to the room. Users of the room (with communication rights) will receive this message.

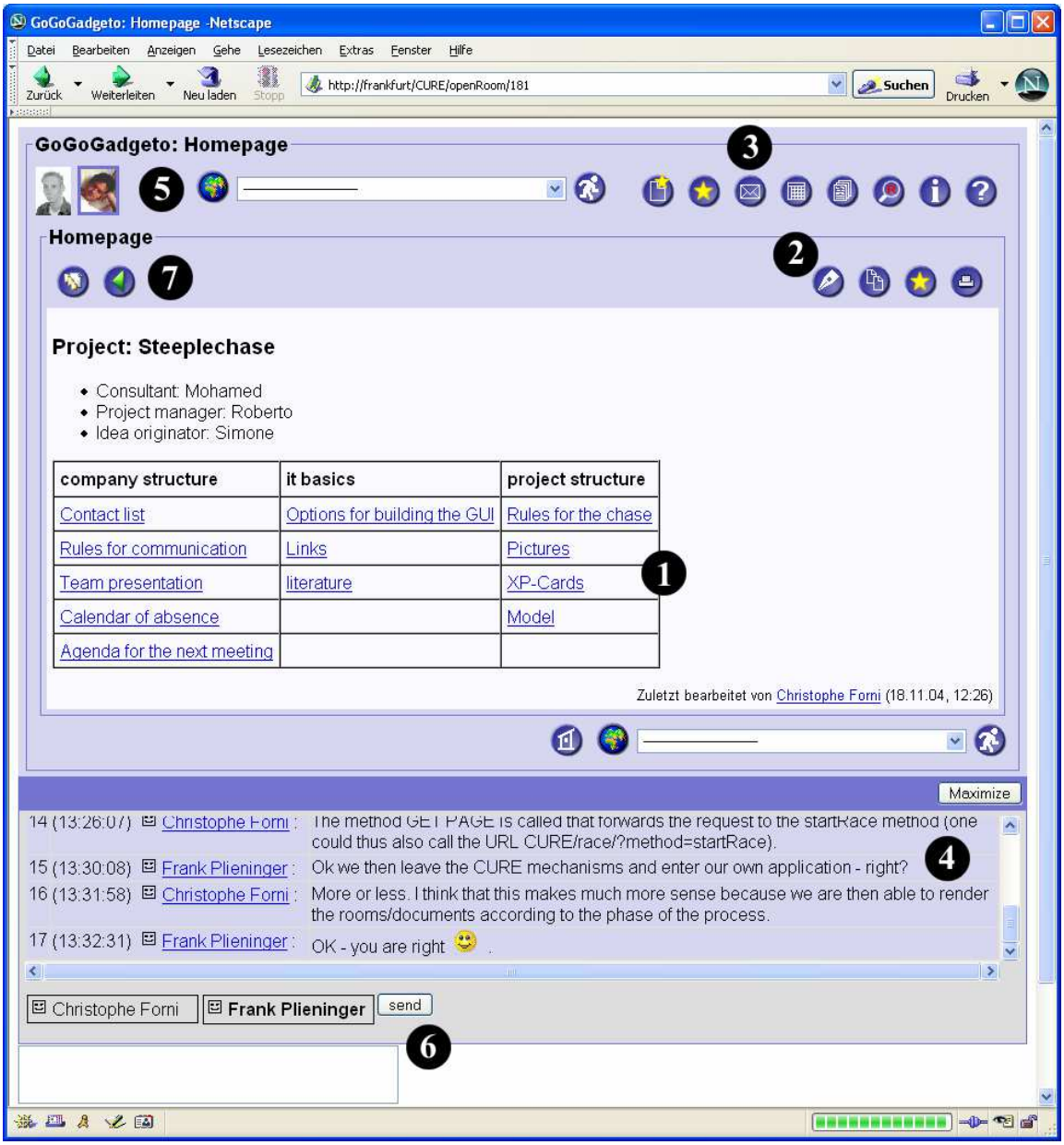

Figure 1: A room in CURE

By providing a plenary room, sharing and communication in a whole class or organization can be supported. By creating new rooms for sub groups and connecting those to the classes' or organization's room, work and collaboration can be flexibly structured. In order to support coordination between the users of a room, different types of awareness information is made available. Firstly, users can see in the room's properties who has access to this room. Secondly, users can see which users are currently in this room (5). Thirdly, if the chat is enabled in the room, they can directly start chatting to each other (6). Fourthly, daily reports automatically posted to all users of a room include all changes made since the last report was sent. If users change pages, the previous state of the page is kept as a version (7) - thus, providing artifact change histories.

To construct structured learning environments, a room may be connected to adjacent rooms, thus forming a virtual learning environment.

Using CURE to support the project method in a distributed lab

In the previous section, we identified several phases for distributed software engineering labs. In the following, we will describe how CURE can be used during these phases. Note that we also show how to use CURE in the co-located phases where we equipped all co-located students with wireless notebooks.

Administration. In CURE, each department of the university can have a public room to organize its courses. For announcing the course, teachers create a new lab room for the lab course. Students are notified by a daily

report of the CURE system that a new room, i.e. course, is offered by the department. The daily report contains a link to the new room. By following this link, students can find out more details about the possible problem space for the lab course. Students interested in taking part in the lab course can then enrol in the lab course by requesting a key for the room in CURE and providing information about their previous experience. CURE informs teachers about key requesting students. Depending on the students' background knowledge, teachers can grant students access to the room as long as the maximum number of participants is not exceeded.

Purposing. In the lab room, students find pages with additional information about the lab course. These pages also describe the students' first task and deadline, which is to create a project outline adhering to the basic requirements for the lab course. Students have to create pages in the lab room describing their individual project outline. After the first deadline, students start discussing the proposed projects either by using the mailing list for the lab room or the lab room's chat. Teachers screen the proposed project outlines to select those ideas matching the learning goals of the lab course best. The selected outlines are announced in the lab room so that the respective students can prepare their talk.

Group building & planning. Each user has a personal home page in CURE, which is used to describe the user to other users. Socializing is not proactively supported by CURE. Thus, it is the teachers' task to direct the socializing efforts according to group characteristics. This is done at the beginning of the first co-located phase. Here, students campaign for their project outline and make their presentation available in the lab room for future reference. After the presentations, students have to form groups. For this purpose, students have to create pages in CURE that describe their personal strengths, interests, and areas needing improvement (comparable to the self image game (McKee, Solas, & Tillmann, 1998, p. 26)). Students browse each others' pages and form well-balanced groups around at least one student interested in project management. To ensure that the groups are well-balanced they refer to the previously created personal pages. After the groups were confirmed by the teachers, each group creates a sub room. They distribute keys to all other group members so that the group can from then on collaborate in their group room.

The first group task is that the group members vote on projects that are interesting for them. Based on the resulting project ranking, the groups elaborate project applications for the most interesting projects on respective pages in their group room. They make the pages available for the teachers. Depending on the project applications, the teachers assign one project to each group.

After assigning the projects, teachers introduce the students to possible methodologies and provide additional readings on a designated literature page in the lab room. Each group agrees on one methodology for the execution phase and on the use of CURE as a vehicle for communication and collaboration, e.g. they may agree on reading mails at least every other day or to meet regularly for a group chat. These agreements are documented in their group room. In a next step, the groups begin to create CURE pages specifying required work packages. They link the pages in a logical order on an index page. Additionally, responsibilities and roles are documented in CURE. After this, the first co-located phase ends and distributed work continues.

Execution. Group members sign up for a work package by adding their name to the respective work package page. While performing the work, they use CURE as a work log. The work logs as well as the CURE daily report are used by the teachers and project leader for group monitoring. Teachers and students discuss milestone results via the mailing list of the group room. If the group does not fulfill requirements, teachers intervene and demand a solution. To support inter-group exchange teachers create topic-centered sub rooms of the lab room and require students to discuss topics of common interest in these rooms.

Judgement. In the final co-located phase, each group creates a public sub room for presenting the project outcome. The presented material serves as a basis for discussing the outcome with the other groups in a co-located plenary meeting. Later on, each group reflects about their project using the persistent work logs in CURE and other artifacts like pages, mails, or chat logs. The insights of the reflection phase are stored as pages in the group's room so that students can later recapture the lessons learned. The final phase ends with a collaborative grading of the group.

EXPERIENCES

In order to evaluate whether teachers and students can successfully employ our approach to run distance lab courses, we observed the usage of the system in two instances of a lab course in the computer science program of our university. In the first course, students had to develop a groupware application for collaborative gaming. The second lab course was about tools supporting collaborative learning. We proposed the eXtreme Programming methodology (XP) (Beck, 1999) in both courses. This is a lightweight process model used successfully in co-located courses (cf. the course in Karlsruhe mentioned in the introduction).

Method

Setting. Students were studying from home or office using the Internet and CURE. Both courses used a form of blended learning. Groups would initially collaborate at a distance, and then meet for 3 days (first iteration) or 4

days (second iteration) at our campus, followed by distance collaboration during the term (approx. 4 months), until a final presentation meeting (2 days) at the campus again. During the distributed phases, teachers from our university did communicate with their students via CURE and sporadic phone calls.
Design. We observed the system usage and co-located interaction in the two lab courses.
Subjects. We did not select subjects on a controlled basis. Rather, we deployed our system in existing courses. Students did enrol in these courses as part of their regular studies. Students at our distance learning university are mostly employed elsewhere and thus studying part-time from home or office. In the first course 34 students worked in 6 groups, in the second course 21 students worked in 3 groups. Teachers are regular professors or teaching staff at our university with experience in e-learning over the Internet (e.g. using Mail, Newsgroups, WWW, BSCW, IRC) and expert knowledge about the CURE collaborative learning environment.
Procedure. In the above two courses, teachers were involved in the development of CURE, and thus were already experts in its functionality. Teachers developed their learning environment (a set of rooms and materials) on their own. Then, teachers prompted students to use the system. Students did not receive any training. Rather, they were pointed to the online system manual, which contained a detailed introductive scenario and a reference section. In order to learn from our experiences of the first course, we adapted the approach where needed and observed the implications of these changes in the second course.
Measures and evaluation infrastructure. Due to the distance learning setting, it is difficult to conduct direct observational studies (e.g. taking videos). Instead, we regularly conducted interviews with the teachers present at our university and with some student groups, when they were present at our campus. We also examined the shared rooms and artefacts created by the groups - including their mailboxes and chat logs.

We observed the use of the system with a focus on the four problems listed in the introduction: motivational, schedule, process, and communication and interaction problems. In addition, we looked at the learning outcome with respect to distributed software development and at their achievements in distributed teamwork.
Measures resulting from interviews are reported as anecdotal evidence.

Results

We analyzed the data collected during both lab courses with respect to the following questions:

1. To what degree did students experience **motivational problems**?
2. To what degree did students experience **schedule problems**? How did they solve them?
3. To what degree did students experience **process problems**? How did they solve them?
4. To what degree did students experience **communication and interaction problems**? Did group building help to alleviate the problems?
5. Did students reach the **domain-oriented and social learning goals** of the lab course? Did they show sufficient mastery of distributed software development and teamwork skills?

Course 1: Design of collaborative games

The first lab course followed the proposed outline in most phases, but there were some differences that proved to fail. We will report on these failures since they motivated the genesis of the final sequence of action, as it was applied it in the second iteration. The main differences were in the group building and planning phase:

- The time spent on social games was very low.
- All students were allowed to present their ideas, not just a selection matching the initial requirements.
- Students were allowed to vote for project ideas and the three winning ideas were used as the only basis for group formation.
- Since the co-located phase lasted only three days, a complete work plan was not requested before the group left the co-located phase.
- The role of project leader was not a requirement of the teachers (i.e. groups could organize freely).

These decisions led us into several problems that were our impetus for adapting the educational method to its current form. We will now report the results regarding the five research questions.
Motivational problems: All 34 participating students submitted their personal project proposal. Although each participant was motivated to think about a personal idea, the discussion of other students' ideas during the purposing phase did not take place to the desired extent. Apart from 5 slightly unmotivated students, they presented well motivated presentations of the project. Six ideas were selected and students instantly formed groups for these ideas. They reported that the task was challenging and interesting. Besides one student, there were no free-riders. The problem of this specific student was caused by his deficient background knowledge. Several groups had deficits in major areas required for solving the task. This disencouraged group members in working on the solution. These were the reasons, why group formation was changed in the second iteration.
Schedule problems: All groups reported problems in their schedule concerning the project planning. This seems to be mainly a result of the rather short first co-located phase. The groups started distributed work without having a concrete work plan and clear milestones. Without the concrete work plan, the groups underestimated the required effort and therefore had enormous problems to finish their task in time. In two cases, this was only

possible due to the exceptional commitment of single group members. Another group only finished their task by organising co-located development sessions in the final third of the project.

Process problems: All groups were introduced to the eXtreme Programming methodology. Although this is a lean methodology, none of the groups was able to follow all principles of XP. This was partially due to the fact that the groups were distributed and XP has no provisions for distributing critical parts (Schümmer & Schümmer, 2001). But the main reason was that the students had no shared understanding of their tasks and thus failed to play the planning game, a way for identifying customer needs and prioritizing development tasks.

Communication and interaction problems: All groups communicated actively using the mailing list of their room. The total amount of exchanges messages ranges from 205 to 411 in the different groups. The groups with the highest communication load had the strongest commitment to roles and areas of expertise. Especially, the project leader was clearly visible. These groups produced the best results.

Several groups reported on members who were not responding for several weeks. Personal problems resulting in the lack of time were the main reason for this behaviour. These members did not communicate their problems to the group. In one case the problems resulted from a wrongly configured spam filter.

Three groups used a framework for implementing the collaborative game. To introduce these groups in the use of the framework, the teachers created a discussion room, which was used by members of all three groups.

One drop-out was caused by problems in the group structure. The team did agree on the use of a specific technology in the co-located phase. In the distributed execution phase, one group member tried to reverse this decision, which caused a very long discussion. He finally left the group as it would not change the decision. Nevertheless, the discussion caused a lack of trust in the group that was not re-established until the end.

Regardless of the above problems, all group members managed to participate in the group process and contributed to the project solution. Except for the one free-rider mentioned above, no group had difficulties in accepting a common grade for all group members.

Learning goals: All groups produced a working solution to their problem. They reached the desired technical learning goals of the development of distributed collaborative applications. With respect to project organization, the outcome was not satisfying for four of six groups. These four groups did not come up with a convincing project structure. Although often learning was by failure, all groups reported that they learned important lessons for teamwork and software development.

Course 2: Development of tools for collaborative learning

The task in this course was that students should develop tools that assist them in collaborative e-Learning – a task that they are frequently facing in their studies. The resulting tool should be integrated in the CURE environment and foster learning in an entertaining way.

The course had 21 participants. Its structure was changed due to the lessons learned from the first iteration so that it matched the educational method presented in section 2 of this paper. The duration of the first co-located phase was extended to 4 days, and lecture-style parts were tightened, e.g. to a pre-selected number of project ideas that were presented in the plenary, or a shorter introduction to base technologies and methodologies.

Motivational problems: Again, all students prepared a project proposal. The elaborateness of these proposals was in all cases higher than in the case of the first iteration. The ideas were screened and eight ideas that had an appropriate level of difficulty were selected. All screened ideas were rated by the teachers as meaningful supportive means for studies by the participating students. But still, the discussion of other students' ideas during the purposing phase did not take place to the desired extent.

Each group created sound project applications for three of the eight selected project ideas. This reflects that group members could identify with most of the presented ideas. Motivation was thus no longer a problem in the initial phase. In the following planning and execution phases this trend persisted.

Schedule problems: The first iteration showed that students should not be allowed to leave the first co-located phase without a concrete work plan. Although teachers provided less detail on the planning processes used in XP or other software development methodologies, all groups created more concrete plans than in the first iteration. This may be due to a larger focus on demanding work packages in the planning phase. The groups agreed to milestones and documented these in CURE. Wrong calculations of efforts or personal schedule problems led to adaptations of milestones. Nevertheless, the groups always maintained a current project plan.

Process problems: All groups came up with a variant of the XP methodology that was shaped to the distributed nature of the team. A special focus was on the planning game. A result of the planning game was that all groups, compared to the groups in the first course, had a much better understanding of their task and therefore were able to agree on very concrete work plans. The planning cards were managed in CURE.

Communication and interaction problems: The changed way of group formation led to a clear understanding of roles and a responsible project lead in each group. The group members assigned themselves to areas of expertise. The groups frequently reflected about role assignments. For instance, in one case, the project leader changed during the first days after the group noticed that other group members provided a higher competence in moderation and structuring of planning discussions. After two more weeks, the project leadership was split in

two parts, a technical and an organizational project leader – since the group discovered that one group member had a very high technical competence. The current role assignment was always documented in the group room in CURE. In another example, the project leader had a severe accident and had to stay in hospital for three weeks. Although he interacted with his group using CURE via the Internet terminal of the hospital, the group compensated his absence by reassigning the tasks. Both examples show that group members were aware of the group's role distribution and adapted their work process during execution.

All groups established communication rules that were documented, presented, and discussed also with other groups. Example rules were as follows:

- *Team members have to read and answer the mails in the CURE room at least every three days.*
- *Vacations have to be communicated within the group room by filling a calendar of absence.*
- *All team members have to meet for a group chat every Friday at 9pm.*

The rules were followed during the course. The communication was more active (with respect to number of exchanged messages and lengths of the chat sessions) than in the first iteration. In one group, the students frequently met in the chat to explain different subject areas to each other. These discussions often had an important social component where team members motivated other team members who had problems understanding technical details. We assume that the more intensive communication is a result of the extended time for socializing and group formation during the first co-located phase.

Learning goals: As in the first course, all groups delivered working results. Additionally, all groups learned more about the process of distributed software development. They acquired social as well as technical practices for computer-mediated distributed project work.

CONCLUSIONS

In this paper, we reported about a blended learning approach that uses the project method in the context of distributed software development lab courses. We showed how the project method can be appropriated to a distributed CSCL setting and how the collaborative virtual learning environment CURE can support this method. From findings of traditional lab courses, we identified several problems when applying the project method in a distributed setting, i.e., motivational, schedule, process, communication and interaction problems. To address these problems, we proposed a combination of the project method with a blended learning setting. In order to evaluate our approach, we conducted two case studies in two distant learning courses at our university. Results suggest that students and teachers can apply the method successfully, that the above problems are reduced, and that not only domain skills (e.g. software development) but also social skills are learned by the students. All groups showed a large level of engagement and did not encounter motivational problems.

Up to now, the project method has been used for fully co-located settings. We are aware of a few settings, where the project method was applied in a fully distributed setting (e.g. by Thomas (2002)). To our knowledge, it was up to now not applied in a blended learning approach (combining distributed and co-located phases). Thus, the proposed modifications of the method, the suggested way of supporting it in a CSCL environment, and the results from two case studies provide new insights into ways for project-oriented CSCL.

In our research we used an iterative approach. During the evaluation of the first version of our approach we still encountered schedule problems. This problem was addressed by extending the first co-located phase in the second course, so that groups now started the distributed work phase with a concrete work plan. In the first lab course we also still encountered process problems. In the second lab course these were addressed by giving groups more time for the planning game so that they left the co-located phase with a much better understanding of their tasks. Since the results of the first lab course suggest that a higher degree of communication in the group leads to better results and that a group needs a mix of students covering all necessary skills to solve the task, we increased the time spent on social games and changed the group formation process. In the second course, students were requested to form groups that are well-balanced according to the available skills rather than based on preferences for project ideas. These modifications of the method proved effective. Although the two iterations provided first insights, future iterations of courses using the project method are needed to show the general applicability of the educational method in a blended learning context.

From a technology perspective, we conclude that CURE seems a good means to support lab courses following the project method. However, despite the use of the same technology (i.e. CURE) the results and encountered problems in both courses differ to a large extent. This emphasizes the role of the social process, which has to complement the technical infrastructure of a CSCL environment. We could observe that problem-centered interaction can be effectively performed in a distributed setting. Socializing and group formation as well as an initial planning of the group process should on the other hand be performed in the co-located phases. Although our second iteration showed to be effective, the ideal balance between co-located and distributed actions is still to be found. Future research is needed to find the optimal fit between social process and technological support.

Besides this question, we are currently experimenting with special groupware tools to support socializing also before the co-located phase. First steps into this direction look promising and we will observe the effects of tools for building up user communities (Schümmer, in press) in future iterations of the lab course.

References

Appelt, W. and Mambrey, P. (1999) Experiences with the BSCW Shared Workspace System as the Backbone of a Virtual Learning Environment for Students, Proceedings of ED-MEDIA99.

Beck, K. (1999). eXtreme Programming explained. Addison Wesley: Reading, MA.

Becking, D., Berkel, T., Betermieux, S., Clossen, M., Feldmann, B., Rademacher, G., and Schlageter, G. (2004) Motivation and Structuring in a Virtual Database Practical by Means of Roleplaying, Proceedings ICDE 2004, Hong Kong.

Bentley, R., Appelt, W., Busbach, U., Hinrichs, E., Kerr, D., Sikkel, K., Trevor, J., and Woetzel, G. (1997) Basic support for cooperative work on the world-wide web. *International Journal of Human-Computer Studies*, 46, 827-846.

Greenberg, S., and Roseman, M. (2002) Using a Room Metaphor to Ease Transitions in Groupware, in Ackermann, M., Pipek, V., Wulf, V. (Eds.): *Beyond Knowledge Management: Sharing Expertise*, MIT Press: Cambridge, MA.

Gudjons, H. (1997) *Handlungsorientiert lehren und lernen. Schüleraktivierung, Selbsttätigkeit, Projektarbeit*. Klinkhardt: Bad Heilbrunn.

Haake, J. M., Schümmer, T., Haake, A., Bourimi, M., and Landgraf, B. (2004) Supporting flexible collaborative distance learning in the CURE platform. Proceedings of HICSS-37, January 5-8, IEEE Press.

Hadim, H. A. and Esche, S. K. (2002) Enhancing the engineering curriculum through project-based learning. In: 32nd ASEE/IEEE Frontiers in Education Conference. IEEE Press: Boston, MA.

Jacobson, I., Booch, G., and Rumbaugh, J. (1999) *The Unified Software Development Process.* Addison-Wesley: Reading, MA.

Kerth, N. L. (2001) *Project Retrospectives: A Handbook for Team Reviews*, Dorset House Publishing.

Kilpatrick, W. H. (1918) The project method. *Teachers College Record,* 19, 319-335.

Knoll, M. (1997) The project method: Its Vocational Education Origin and International Development. In: *Journal of Industrial Teacher Education.* **34** (3).

Kölbl, D. (2001) Ordinance governing the principles of project-centred teaching. Austrian Federal Ministry of Education, Science and Culture: Vienna. http://www.bmbwk.gv.at/medienpool/6788/ProjeE.pdf

Leuf, B. and Cunningham, W. (2001) *The WIKI way*. Addison-Wesley, Boston, MA, USA.

McKee, N., Solas, M., and Tillmann, H. (Eds.) (1998) Games and Exercises – a manual for facilitators and trainers involved in participatory group events. UNICEF-ESARO: Nairobi, Kenya.

Pfister, H. R., Wessner, M., Beck-Wilson, J., Miao. Y, and Steinmetz, R. (1998). Rooms, protocols, and nets: metaphors for computer-supported cooperative learning of distributed groups. Proceedings of ICLS-98, Georgia Tech, Atlanta, 242-248.

Schroeder, U., Brunner, M., and Deneke, M. (1998) Constructionist Learning in Software Engineering Projects. in P. Klint, J. Nawrocki: International Software Engineering Education Symposium, Poznan, Poland, Scientific Publishers OWN.

Schümmer, T. and Schümmer, J. (2001) Support for Distributed Teams in eXtreme Programming. In Succi, G., Marchesi, M. (eds.): *eXtreme Programming Examined.* Addison Wesley: Reading, MA.

Schümmer, T. (in press). Patterns for Building Communities in Collaborative Systems. In: Proceedings of the 9th European Conference on Pattern Languages and Programs, UVK: Irsee, Germany (in press).

Thomas, W. R. (2002). An analysis of student collaboration and task completion through project-based learning in a web-supported undergraduate course. Louisiana State University.

Tuckman, B.W. (1965) Developmental sequence in small groups. *Psychological Bulletin*, **63**, 6, 384-399.

Williams, L. and Upchurch, R. L. (2001) In support of student pair-programming. In Walker, H., McCauley, R., Gersting, J., Russell, I. (eds.): Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education. ACM Press: New York.

Woods, D. (1994) *Problem Based Learning: How to Get the Most from PBL*, McMaster University.

Yoon, G. (2001) A study of project-based instructional systems design (p-isd) model development for engineering practice education. In: International Conference on Engineering Education. INEER: Oslo, Norway. http://www.ineer.org/Events/ICEE2001/Proceedings/papers/444.pdf.