Software Engineering Students meet Interdisciplinary Project work and Art

Letizia Jaccheri and Guttorm Sindre
Department of Computer and Information Science
Norwegian University of Science and Technology
7491 Trondheim, Norway
letizia@idi.ntnu.no

Abstract

Do software engineering students need interdisciplinary skills? Do students learn different things from an interdisciplinary project work than from software development projects? How can a specific interdisciplinary project course be organized? This paper provides reflections about these questions, based on the experience gained through running a project-based interdisciplinary course thrice. This course is part of the master degree education at the NTNU, Trondheim, Norway. Students taking the course work in teams of five, often coming from study programs of quite different disciplines. There is no predefined project assignment, instead the teacher has only described an open-ended theme, within which different student teams then define their own project assignment. The paper provides some reflections and lessons learned that can be exploited for designing similar interdisciplinary team project courses in other universities.

1 Introduction

Software projects are by nature interdisciplinary, drawing on many different types of skills and knowledge, both IT-related (e.g., project management, analysis and design, user interfaces, coding, testing, ...) and non-IT (e.g., knowledge of the application area for the software, say, accounting, healthcare, or the arts). The rapid technological change means that career paths are more uncertain for today's software engineering students than they were before, and that the importance of interdisciplinary skills is likely to be growing [12, 14]. This can be addressed in study programmes in two different ways:

- making interdisciplinary study programmes drawing on knowledge from two or more established disciplines, e.g., bioinformatics, computer linguistics.
- educating students that are still experts in one discipline, but give them the skills needed to work together with experts from other disciplines.

It is the second approach which is of interest to this paper. In software engineering education, one way to strengthen the students' abilities to work with people from other disciplines would be through team projects:

- The problem to be solved by the project team may demand knowledge from several disciplines. In a software project, the typical combination would be IT + one or more application-oriented disciplines, depending on the customer for which the software is intended (whether that customer is real or invented).
- Even more ambitiously, the project may necessitate communication with people from other disciplines, or even compose teams with students from different disciplines.

Yet, while most contemporary university level software engineering programmes include one or more team projects, it seems that interdisciplinarity is rarely a strongly emphasized learning goal for such projects. Part of the reason might be that the core software engineering skills are themselves quite a bit to master, requiring team projects of substantial size to be properly addressed – a too small project assignment can easily be solved by ad hoc coding, not needing any planned process or team collaboration. Hence, adding interdisciplinary challenges on top of an already ambitious team project may be too much in terms of student workload and pedagogical risk.

At the NTNU, we expose our software engineering students to three substantial team projects during the course of study, two with a focus on core software engineering skills, and the third instead focussing on learning goals related to interdisciplinarity. This third project course is called *Experts in team* (EiT), and is compulsory not only for IT students but also from students of other programs within our university. This paper presents and evaluates our experiences after being involved in three subsequent offerings of this course.

The rest of the paper is structured as follows: Section 2 shows how the EiT course complements other team projects offered to the software engineering students, and explains why it needs to be placed late in the course of study. Section 3 describes the EiT course in general, while section 4 goes into more detail about the particular project instances the first author has been supervising. Section 5 presents research questions, research method, and evaluation results, both for our particular project instances and for the EiT course in general. Section 6 then reviews related work and discusses how our project is different from what is reported by others. Section 7 concludes the paper with some lessons learnt and indications for future work.

2 Three complementary team projects

At the NTNU, as in most other contemporary software engineering study programmes, the students have several team projects to train various aspects of the software development process. Disregarding smaller projects within single theory courses[13], three team projects are compulsory to all our software engineering students:

- a software engineering project in the second year [18], where teams of 4-5 students start out with teacher-supplied requirements to perform the design (UML), coding (Java), and testing.
- a systems development project in the fourth year (Autumn, i.e., 7th semester) [2], where each team of 6-7 students is assigned to a real customer, for whom they have to do problem analysis and requirements engineering, design and a partial implementation of the system.
- the EiT project (4th year Spring, i.e., 8th semester), focussing on interdisciplinarity and self-reflection, as will be further described below.

These three projects can be seen to complement each other. The first focuses on late development phases, while the second on early phases, but both with a strong intra-disciplinary focus on core software engineering skills. The EiT project is significantly different in that:

- the team members are no longer all IT students but drawn from several different study programs
- as long as they stay within a defined theme, the students are free to conceive their own project assignment, whereas requirements were given by the teacher in the first project and elicited from the customer in the second. Nor is any specific project process imposed on the teams, who can define whatever milestones they like.
- there are no specific demands to develop software (although the team may do so if they wish). This reflects the important recognition that software is not a goal in itself: What a potential customer wants is to have a problem solved, or to gain useful knowledge, and if this can be achieved better without software, so be it.

Still, the software engineering (SE) students are supposed to use their expertise as software engineers in their EiT project. Hence, each team where one or more members are software engineering students need to define a project assignment for themselves that, if not developing software, at least lets their SE background come into play. Similarly, the problem should also require biological knowledge for its solution if a Biology student were in the team, or knowledge of art if an Art student were in the team. Thus also the motivation for the late placement of EiT in the course sequence: The students should already be ”experts” in their field of study when undertaking the EiT project, thus being challenged to see how their expertise can be made useful in collaboration with other students with a quite different expertise. The next two sections will explain in more detail how the course is run.

3 EiT: the course in general

Experts in team (EiT) is a course of 7.5 ECTS credits [1], compulsory for almost all 4th year master students at the NTNU. Students work in interdisciplinary teams and are asked to define a problem description and establish a project to solve this problem. There are four general learning goals:

[1] European Credit Transfer System, 60 credits per year, i.e., EiT will be one of four courses that a full-time student takes during the Spring semester.

LG1: acquire an understanding of own competence and how it may be used for the benefit of the team.

LG2: gain experience in teamwork as a means to solve interdisciplinary problems.

LG3: acquire insight in own behavior and how it influences collaboration in the team.

LG4: acquire insight in how one is influenced by the team work.

Each group has to deliver a product report and a process report, counting 60% and 40% towards the grade, respectively. The product report must present and discuss the interdisciplinary problem solution and the scientific methods that have been used to come to the solution. The process report must describe the team process, e.g., how the team cooperated, roles of different team members, whether there were any significant events during the process (e.g., conflicts, how these were solved), and how these can be related to group process theory.

At the time of writing (2006) EiT is taken by 1500 students divided in approximately 50 classes (or villages) of 30 students each, who are composed into 6 teams of 5 students. Each village is supervised by a professor, who has described a fairly open ended theme for that village. Each student team may then invent their own project assignment, and set their own milestones, as long as they stay within the given thematic area and end up delivering the required reports. This openness of the assignment is supposed to foster student creativity and a strong sense of ownership of the conceived project. Moreover, the open assignments make it easier for each team to define a project where every member is able to contribute their own expertise, regardless of which study program they come from.

Nevertheless, it must be admitted that many village themes do imply some backgrounds as essential to the village, while others may be harder to accomodate. For instance, our village on Art and Software could definitely be of interest to Art students (or other closely related programmes like Design or Architecture) and IT students, but it might be harder for, say, a Medicine student to capitalize on her expertise within such a theme. Similarly, the village with the theme "How to pump up more oil from the subsea field Gullfaks?" might easily involve a number of science and technology disciplines, as well as economy, but it would be hard to come up with a project within this theme that would be challenge the expertise of a student of Religion, Drama or French language. Hence, students are not assigned to villages at random, rather each student will make a prioritized application for 5 villages where the student believes that her background will be relevant.

The project is usually run in the Spring semester over 14 weeks. Wednesdays are reserved for village day which means that no courses taken by 4th year students are allowed to schedule lectures on that weekday. This is necessary for the students from different campuses and study programs to be able to meet conveniently. Each Wednesday, a meeting is held in each village, this is compulsory for all students.

Students themselves carry the responsibility for convening, meeting agenda, and leading the meetings. At the meeting, shared problems are discussed to agree on actions to be taken. Some meetings are reserved for technical or team process presentations by the students in accordance with a milestone plan that the students have worked out themselves at the beginning of the semester. By working in EiT where each team member initially has different perspectives on the problem at hand, the students will develop attitudes and interdisciplinary teamwork. In solving a problem that challenges their area of expertise, they will be trained in using their subject skill to contribute to the mutual problem solving process. Through this process the students will be exposed to the challenge in interdisciplinary communication, learn to operate within an interdisciplinary environment, learn to understand the interaction between each member of the team, and learn how this interaction affects them.

The teacher encourages the students to organize their reported results according to the learning goals. LG1 (about own competence) and LG2 (ability to solve interdisciplinary problems) are usually discussed as part of the product report while LG3 (insight in own behavior) and LG4 (insight in team dynamics) as part of the process report. Different groups are free to address the learning goals in their own preferred ways, but there exist some tests that the facilitator team offers to students who want help to reflect over LG3 and LG4.

In the next section, we look at one such village in more detail, with a theme circling around Art and Software. A more general description of the pedagogical framework of the course can be found in [19, 20], while [15] gives an account of another EiT village, whose theme circled around various applications of microelectronics.

4 EiT: The Art and Software village

As all other villages, the Art and Software village has to comply with the general framework

of the EiT course. Hence, each student team is encouraged to identify their own problem within the scope of the village theme – and such that the expertise of each team member is relevant to the problem. Also each team will define their own goals and project plan (with or without milestones on the way), and the final product and process reports will count 60/40 towards the course grade. The village has been run three times, with slightly changing theme titles:

- in 2004 the name was *Art and IT*, and the village attracted 9 CS/SE students and 16 students from other study programmes. From this 6 teams were composed, each with 1-2 CS/SE students.
- in 2005 the name was *Art and Technology*, and the village attracted 5 CS/SE students and 19 students from other programmes. From this 5 teams were composed, each with 1 CS/SE student.
- in 2006 the name was *Art and Software*. The village now attracted 13 CS/SE students but only 3 students from other programmes. While the 2004 and 2005 villages were taught in Norwegian, the 2006 village was taught in English.

A first observation to be drawn from this, is that the choice of theme and title for the village might seem to have a strong impact on the students' choices. *Art and Technology* may have felt too broad for CS/SE students, who then sought other villages instead. On the other hand, the more specific *Art and Software* brought the number of CS/SE students to an all time high, but the number of other students was very low. Still, it should be noted that the name alone is probably not the entire reason for this fluctuation in popularity. Students are allocated to the various villages by the central EiT staff, based on a bidding process where each student submits a prioritized list of village preferences. Thus, the number of students per village will also depend on the type and total number of villages offered. One notable point here is that EiT was initially strongly dominated by the technology study programmes, but from 2004 to 2006 there has been a gradual increase in villages offered from Arts and Humanities teachers, causing a stronger competition for students who might have been particularly interested in the Art aspects of the village. Moreover, the switch from Norwegian to English as the teaching language of the village (meaning that reports also had to be written in English) may have scared away many Arts and Humanities students, for whom the readings may have been mainly in Norwegian in their previous courses, while CS/SE students are accustomed to course literature in English from their freshman year. Anyway, the 2006 student composition for the village reveals that the allocation procedure of the central course staff did not function according to the intention, as this time around the CS/SE students mainly had to collaborate with other students of their own discipline.

In the following, we will look at the three subsequent project experiences in more detail.

4.1 The 2004 village: Art and IT

This year, most of the groups (5 out of 6) chose to work for an external customer. Three groups developed software for the project LIVE_LIFE, suggested by the Norwegian artist Espen Gangvik. The idea of the artist is to develop an animated picture that changes over time. The picture does not interact with its environment. The life cycle of the picture could be as long as one wishes. Theoretically the observer should be able to follow the evolution of the picture during his/her whole life. The three teams produced three different systems according to the initial specification and the cooperation with the artist. Two other groups worked with the artist Kristin Bergaust, one with tracing for artistic purposes, and one developing an interactive art installation. Only one group chose not to work with a customer, thus defining a project idea totally on their own. This group wrote a report about a 3D engine for art. This group was, incidentally, the only one not to develop any code in their project, their report only dealing with the analysis and design stage.

4.2 The 2005 village: Art and Technology

In 2005, none of the teams had specific customers. This was not due to any dissatisfaction with the customers of the previous year, but more a coincidence. It is not up to the teacher but to each respective team to decide whether they will work for a customer or not, although the teacher could provide advice on the pros and cons of each alternative, as well as obtain contacts with artists willing to be customers. One group made a futuristic collection of pictures and sounds, another a flash animation parodying three well-known TV reality series. Two groups worked on tools, one supporting photo manipulation for artistic purposes, the other (report ending with a design, no code) a framework for photography teaching in schools. Finally, one group delivered a more essay-oriented product report, discussing parallels between art

and technology. Hence, three of the five groups developed code.

4.3 The 2006 village: Art and Software

In 2006, one group worked for a customer, the artist Oyvind Brandsegg. They developed a software component for a music installation which takes care of the transmission of music between the server, the physical installation, and the WEB interface of the system (http://www.flyndresang.no/). The other groups developed an interactive installation and a software tool (Artware), respectively. Artware supports WEB based community building based on sharing and development of pictures. This year, all the teams developed code in their projects, which is perhaps not surprising given the huge overweight of CS/SE students in the 2006 village.

5 Evaluation and discussion

5.1 Research Method

The main research method used in the evaluation and gradual improvement of the EiT village about Art and IT/Technology/Software has been that of *action research* [21], which is a quite popular research method in the field of education [8]. With such an approach, the research goes through cycles of four stages:

- *plan* the project course
- *act*, i.e., run the project as facilitating teacher
- *observe*, e.g., what the students do in their projects, how the project teams function, to what extent they are satisfied, and to what extent the learning goals of the course seem to be reached.
- *reflect* upon the teaching practice and course structure, i.e., which features of the project led to success or failure for the various teams' course satisfaction and learning outcomes? The result of this reflection should be ideas for improvement, which are subsequently input to the next cycle, i.e., the planning stage for next year's project.

So far, this cycle has been run three times for the project village under discussion. The main sources of observation have been the following:

- direct observation of the students during their teamwork
- oral and email consultations with the students
- input from teaching assistants
- reading the product and process reports of the teams, as well as observing any products beside the reports (e.g., art pieces, software demos)
- student feedback on questionnaires specifically related to the particular village

Additionally, the staff centrally responsible for the EiT course for the entire university evaluates the course through a more general questionnaire distributed to all the 1500 students taking the course. But this survey only yields gross averages for a huge number of students (of which only about 2 percent belong to our particular project village), so it is hard to draw concrete guidance for improving the particular village from these data.

A particular challenge for the reflection and gradual improvement of the project village is that the teacher is not free to make changes to the project structure, as this must always be in compliance with what is decided by the central EiT staff. So, for instance, the teacher would not be allowed to change the 60/40 weighting between the product and process report, or to deviate from the centrally formulated learning goals for the course.

5.2 Research questions

These three years of experience has given input to address several research questions:

- questions concerning the general framework of EiT, which is beyond the control of the individual village teacher / facilitator, such as:
  - RQ1: does the EiT course function as intended? I.e., are the learning goals reached, and is it a satisfactory learning experience for the students?
  - RQ2: do the students learn different things from the EiT project than from their earlier software development projects, i.e., is EiT a useful supplement for the software students rather than just more of the same stuff?
  - RQ3: how should the overall framework of the course be changed to improve the course further?
  - RQ4: what is the ideal composition of student teams, with respect to the number of different disciplines involved?
- questions concerning the role as facilitator, such as:

- RQ5: what are the advantages and disadvantages of a generic vs. specific village theme description? How could the theme description be improved for future villages to attract more students?
- RQ6: how is facilitation in an open-ended project like this different from teaching a stricter software development project course? What are the advantages and disadvantages of various styles of facilitation? (e.g., being active in providing advice, or being more laid-back, waiting until the students run into trouble and actively seek advice). How could the practice of facilitation be improved for future villages?
- RQ7: what are the advantages and disadvantages for teams working with customers vs. those who do not? If working with a customer, to what extent should the facilitator instruct the customer to ensure an optimal learning experience for the students?

In the following, these research questions will be addressed based on various findings and experiences from the offerings of the village so far.

5.3 Reflection

We use our research questions as a tool to structure our reflections about the course:

- RQ1: in order to evaluate if the EiT course function as intended, one should run systematic validation of the learning level achieved by students with respect to the learning goals.

  Learning goals function well to stimulate oral dialog with students. This is reflected in the process documents written by the students which often refer to these discussions. Learning goals guide students in their learning process and this is reflected in the process and product reports. The distinction between process and product is fundamental to make students aware of the existence and importance of the learning and inter personal process. On the other hand, the EiT learning goals are difficult to be converted into measurable properties.

  Almost all groups state in their process report that if they could start once again they would use more time at the beginning to know each other and to establish themselves as a team. Some groups complain that there is not enough interaction at village level and that the teacher should devote more energy to let students know each other at village level.

- RQ2: EiT differs from other courses and project courses as in EiT the goal for the students is not to acquire new knowledge but to get insight in own knowledge and to apply it to new contexts. From discussions with students, from our observations of students discussing with each other and with students assistants, we acknowledge that students strive and progress toward elicitation and recognition of their own competence. Again, it is difficult to quantify this notion of maturation with respect of insight in ones own knowledge. By reading the documents that students have produced, we can use the references to specific literature as a measure of competence.

- RQ3: it is important that EiT evolves to reflect the lessons learnt in the different villages. First of all, it is important that customers, teachers, and students be able to provide feedbacks in a way that the course evaluation is formal and well understood by everybody and that this evaluation can function as an improvement tool. A challenge here is that as EiT becomes bigger and bigger (the number of students and village professors increase each year) and wider and wider (more study programmes join EiT each year) it becomes more bureaucratic and there is less opportunity for the single teacher to influence the general framework.

- RQ4: based on the experience in the art and software village, we claim that the ideal multidisciplinary team must be composed by a combination of computer science students and art and music students. In addition it is beneficial to have students who are neither CS nor art or music ones who can function as inspiring actors in the projects.

  On the other hand, the composition of student teams is implemented by the central administration of EiT and it based on the 5 preferences that students give to villages. Teachers do not have any influence on this process. While the process of allocating 1500 students to 50 villages need some central management, we claim that teachers should be involved in this process. This claim is based on our experience, based on discussions with students and on the comparison between students' knowledge and village theme. It is not feasible that one single administrative person gets enough

insight on 50 village themes to judge if a configuration of students can or cannot be allocated to it.

- RQ5: a good interdisciplinary theme is important to recruit students with different backgrounds.

  To our knowledge there are no well defined criteria to judge the degree of interdisciplinarity of a theme. The quality of a village theme depends on the skills of the teacher who defines it as well as on the background of the students that will be allocated to the village.

  We claim that the theme art and software is of interdisciplinary nature and it is attractive for students from computer science and software engineering, media science, history of art, and design. This claim is grounded in the experience of 2004 and 2005 in which our village was one of the fifty villages with most heterogeneous students. We have changed the theme from IT to technology and then to software.

  As observed by [1] liberal arts topics ("open-ended topics") in which analysis, discussion and interpretation are core competencies and programming issues ("closed, absolute topics") represent the dimensions of freedom and constraints and renders the theme both interdisciplinary and attractive to students.

  Our experience with running the village thrice tells us that both a set of questions (e.g. "How did software change art in the past and how will it be in the future?") and a set of references to resources (e.g., reference to the Trondheim Electronic Art Center) are a good tool to structure a village theme around.

- RQ6: facilitation in an open-ended project requires different capabilities than class room teaching. While the communication in traditional class room settings is mainly mono directional from teacher to students, facilitation requires that the teacher becomes a listener. The teacher-as-facilitator must let the students pose their own questions and seek their own answers rather than providing answers or even hinting what are the most important questions to ask. Facilitation poses different psychological challenges to the teacher. While classroom teaching requires the teacher to remember and be able to explain concepts and theories, facilitation requires the teacher to be able to understand what students discuss and intervene when it is strictly necessary.

  The facilitator must be able to decide from time to time if being active in providing advice, or being more laid-back, waiting until the students run into trouble and actively seek advice. Practices of facilitation can be improved for future villages by letting teachers exchange experience between each other.

- RQ7: customers are valuable to have in the village, but on the other hand they may pose requirements that bereave the students of the responsibility of creating their own project. This is especially true if the customers want to exploit students for short-term goals, e.g., using the students mainly as an implementation resource. In the 2004 iteration an artist dominated the village and made the products become quite interesting (one was displayed at an art exhibition in town). Another potential problem with customers are that they are very busy. In the specific case of our villages, customers are artists who love their ideas and who do not have time to grasp the complexity of the EiT framework.

6 Related work

As mentioned in the introduction, most software engineering programs offer team project courses of some kind, as recommended by [5]. Quite a number of experience papers have been published about such courses, as well as more general papers containing recommendations for how to teach projects. For example, [24] describes a framework for teaching software project courses. [6] describes an approach to teaching in a simulated industrial environment. The topic of educating software engineers from an industry point of view is addressed in [9]. Most software engineering projects have only a limited focus on interdisciplinarity. [25] presents a project course where teams were composed of students from two different universities, one in Finland and one in Russia, the work partly being done by remote collaboration. Since the profiles of the two study programs were quite different, the Finnish students were stronger in software engineering, while the Russians were stronger in Maths. Yet, all the students belonged to their respective CS departments, so the challenge was more one of multicultural teams collaborating remotely than of bridging a disciplinary gap. [7] presents a course curriculum that integrates computer game engineering with software engineering in a project-based learning environment. [22] is about a course implemented by interdisciplinary student projects in the area of computer and video games. However, the discipline gap between computer game engineering and software engineering is limited.

[4] discusses a project couse more similar to ours, teaming together students from several study programs. To teach students about simultaneous engineering, each team included students from three different programs: electrical engineering, manufacturing, and industrial technology. The project assignments appear to be defined in more detail by the teaching staff than what is the case for our loosely defined themes. Accordingly, their projects also had a stronger focus on the delivered product, and somewhat less focus on an explicit discussion of the process and how it was affected by the team's interdisciplinarity. Also, their course did not involve software students, but there was some low-level programming (e.g., of microcontrollers, actuators etc.). [16] describes a project course teaming together majors in computer science and computer engineering. A project with somewhat more disparate disciplines is reported by [3], describing an e-commerce project with teams involving CS students and business students. Similarly, [11] presents a course combining students from Master programs in Audio/Video Production, Computer Graphics, Technical Communication, and Web Design. In all these cases, knowing exactly which study programs the students will come from makes it possible to define assignments that fit this mix perfectly, with a premeditated work-division between the various types of students. In EiT, on the other hand, the assignment must be more loosely defined because, in principle, students from any 5 study programs may end up in a team together.

There are also some relevant publications focusing on the combination of software and art in particular. [23] discusses the role of programmers in interactive art projects, and what must be done to ensure a constructive rather than obstructive contribution from the programmer. But the focus of that paper is not on education, rather on workplace programmers assisting artists. [26] describes a course which has some similarities to ours: it teams together art students and IT students, and it has a strong focus on teaching interdisciplinary collaboration skills. But the teaching took place in the context of a course in VR art, which included ordinary lectures and readings, in addition to several assignments during the term. Hence, the assignments were more more set than in our case, where the one open-ended team project *is* the course. Similarly, [10] presents a team project course where CS students and art students work together, again this took place in a specifically designed course which also involved lectures, partly by a CS teacher and partly by an Art teacher. And again, there are several assignments in the course, not just one. Still the main team project lasted for 12 weeks, and was fairly loosely described (each team is to make an animation). Yet, this means that the wanted product is more defined in this course than in EiT.

Issues related to liberal arts students and computer science education are more generally discussed in [1]. Here, liberal arts topics are regarded as "open-ended" in which analysis, discussion and interpretation are core competencies while programming issues as "closed and absolute".

[17] reports about a case study in which the cooperative knowledge acquisition process is supported and made explicit by use of learning technology. Here, the authors recognize the importance of the the social process and interaction with teachers and peers as a means to reach a higher degree of understanding. [14] discusses *discourse* as a tool for enhancing multidisciplinary skills. However, these publications do not address project courses.

7 Conclusions

This paper has reported on a course called Experts in Teams (EiT), which is really a course framework where a number of different professors (playing roles as 'village chiefs') describe open-ended themes within which student teams then define their own project assignments. Teams are composed of 4th year students from different study programmes, and with the goal that each student should be able to use his/her competence actively in the solution to the team's chosen problem. The course also has strong focus on self-reflection both for teams and individuals. This paper has taken an action research perspective, based on experiences with one particular EiT village that has now been run for 3 years, with a theme where art meets software technology. Main findings are that such an interdisciplinary course gives software students learning outcomes that are quite different from what they get from more traditional software engineering team projects, in particular concerning interdisciplinary skills and self insight. Also, the fact that there is no given project assignment, only a vaguely formulated theme, poses a stronger challenge towards student creativity, and most students have been very satisfied with the discussed village. Yet, findings also indicate that there is a danger of the EiT course as a whole becoming too bureaucratic, and that the requirement to abide by centrally defined learning goals and course structures makes it difficult for the individual teachers to use the experiences gained to improve the students' learning experience from one year to the next.

As further work it could be interesting to establish more performance-oriented tests (e.g., an

experimental pretest-posttest design) whether the learning goals of the course are really met, although this is a considerable challenge since the learning goals are so vaguely formulated that their achievement is hard to measure. It might also be interesting to scale up the project course to a multi-university context as that addressed in [6]. In some cases this could make it easier to ensure enough interdisciplinarity among students in every project, for instance if our university, which has a surplus of technology students, could team up with another university with a different student composition.

References

[1] Peter Bøgh Andersen, Jens Bennedsen, Steffen Brandorff, Michael E. Caspersen, and Jesper Mosegaard. Teaching Programming to Liberal Arts Students: a Narrative Media Approach. In *ITiCSE '03: Proceedings of the 8th Annual Conference on Innovation and Technology in Computer Science Education*, pages 109–113, New York, NY, USA, 2003. ACM Press.

[2] Rudolf Andersen, Reidar Conradi, John Krogstie, Guttorm Sindre, and Arne Sølvberg. Project Courses at the NTH: 20 years of Experience. In *J. L. Diaz-Herrera (ed.): 7th Conference on Software Engineering Education (CSEE'7)*, pages 177–188, San Antonio, USA, January 1994. Springer Verlag LNCS 750.

[3] Karen Anewalt. Utilizing interdisciplinary teams in teaching e-commerce. *Journal of Computing Sciences in Colleges*, 19(2):288–296, 2003.

[4] Abul K. M. Azad, Andrew Otieno, and Radha Balmularikrishna. Interdisciplinary Student Projects toward Simultaneous Engineering: Learning and Issues brought forward. In *Proc. American Society for Engineering Education, 2003 IL/IN Sectional Conference*, Valparaiso, IN, USA, April 2003.

[5] D. Bagert, T. Hilburn, G. Hislop, M. Lutz, M. McCracken, and S. Mengel. Guidelines for Software Engineering Education, 1999.

[6] Lisa J. Burnell, John W. Priest, and John R. Durrett. Teaching Distributed Multidisciplinary Software Development. *IEEE Softw.*, 19(5):86–93, 2002.

[7] Kajal Claypool and Mark Claypool. Teaching Software Engineering through Game Design. In *ITiCSE '05: Proceedings of the 10th annual SIGCSE conference on Innovation and Technology in Computer Science Education*, pages 123–127, New York, NY, USA, 2005. ACM Press.

[8] Louis Cohen, Lawrence Manion, and Keith Morrison. *Research Methods in Education*. RoutledgeFalmer, New York, 2000.

[9] Richard Conn. Developing Software Engineers at the C-130J Software Factory. *IEEE Softw.*, 19(5):25–29, 2002.

[10] D. Ebert and D. Bailey. A Collaborative and Interdisciplinary Computer Animation Course. *Computer Graphics*, 34(3):22–26, August 2000.

[11] Pamela S. Ecker, Jason Caudill, David Hoctor, and Colleen Meyer. Implementing an Interdisciplinary Capstone Course for Associate Degree Information Technology Programs. In *Proc 5th conference on Information technology education*, pages 60–65, Salt Lake City, UT, USA, 2004. ACM Press.

[12] Atila Ertas, Timothy Maxwell, Vicky P. Rainey, and Murat M. Tanik. Transformation of Higher Education: The Transdisciplinary Approach in Engineering. *IEEE Transactions on Education*, 46(2):289–295, May 2003.

[13] M. Letizia Jaccheri. Software quality and software process improvement course based on interaction with the local software industry. *Computer Applications in Engineering Education*, 9(4):265–272, 2001.

[14] Y. Julliard and A.W. Schwab. The Role of Discourses in Multidisciplinarity. In *Proc. International Symposium on Technology and Society (ISTAS'01)*, page 0007, Stamford, CT, USA, July 2001. IEEE.

[15] Bjørn Larsen. Experts in team, interdisciplinary project. In *Proc. 2005 IEEE International Conference on Microelectronic Systems Education (MSE'05)*, Anaheim, CA, USA, June 2005.

[16] S. Mosiman and C. Hiemcke. Interdisciplinary capstone group project: Designing autonomous race vehicles. In *Proc. 31st SIGCSE technical symposium on Computer Science Education (SIGCSE'00)*, Austin, TX, USA, March 2000.

[17] Ekaterina Prasolova-Forland and Monica Divitini. Supporting collaborative construction of knowledge: Lessons learned. In *V.*

Uskov (ed.): Proc. 8th IASTED International Conference on Computers and Advanced Technology in Education, CATE 2005, pages 147–152, 2005.

[18] Guttorm Sindre, Tor Stålhane, Gunnar Brataas, and Reidar Conradi. The cross-course software engineering project at the NTNU: 4 years of experience. In *Proc. 16th International Conference in Software Engineering Education and Training (CSEET'03)*, Madrid, Spain, March 2003.

[19] Bjørn Sortland. Experts-in-team - multi-disciplinary project. In *Proc UNIQUAL, 2nd International Conference on Universities' Quality Development*, Kaunas, Lithuania, October 2004.

[20] Bjørn Sortland. Interdisciplinary Team-Work: Preparing Students for Work-Life. In *Proc. SEFI 2005*, Ankara, Turkey, September 2005.

[21] Ernest T. Stringer. *Action Research*. SAGE Publications, Thousand Oaks, CA, 1999.

[22] Elizabeth Sweedyk and Robert M. Keller. Fun and Games: a new Software Engineering Course. In *ITiCSE '05: Proceedings of the 10th annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, pages 138–142, New York, NY, USA, 2005. ACM Press.

[23] Greg Turner, Ernest Edmonds, and Alastair Weakley. Seeing Eye-to-Eye: Supportive Transdisciplinary Environments for Interactive Art. In *Proc 9th International Conference on Information Visualization (IV'05)*, pages 912–919, London, UK, July 2005. IEEE Press.

[24] David A. Umphress, T. Dean Hendrix, and James H. Cross II. Software Process in the Classroom: The Capstone Project Experience. *IEEE Software*, 19(5):78–85, 2002.

[25] A. Inkeri Verkamo, Juha Taina, Turjo Tuohiniemi, Yury Bogoyavlenskiy, and Dimitry Korzun. Distributed Cross-Cultural Student Software Project: A Case Study. In *CSEET '05: Proceedings of the 18th Conference on Software Engineering Education & Training*, pages 207–214, Washington, DC, USA, 2005. IEEE Computer Society.

[26] Guy W. Zimmerman and Dena E. Eber. When Worlds Collide! An Interdisciplinary Course in Virtual-Reality Art. In *Proc. 32nd SIGCSE technical symposium on Computer Science Education (SIGCSE'01)*, pages 75–79, Charlotte, NC, USA, 2001. ACM Press.