FACTA UNIVERSITATIS (NIŠ)
SER.: ELEC. ENERG. vol. 18, No. 2, August 2005, 165-180

Project-Based Learning in Student Teams in Computer Science Education

Andreas Breiter, Görschwin Fey, and Rolf Drechsler

Abstract: Designing information systems according to user requirements is crucial for software developers. In computer science education, acquiring necessary social skills to elicit and define those requirements is underdeveloped. We introduce a student-centered, project-based learning approach with a student team project, which tries to support these learning processes. Based on existing examples for project-based learning in other disciplines, the didactical concept and the integration into the curriculum are explained. Using two exemplary student team projects, the core learning processes are described. This approach allows students to explore methods for project management as well as requirements analysis and participatory design with real end-users. The results of the project according to student evaluation are presented and conclusions about the value added of student team projects for computer science education are drawn.

Keywords: Computer science education, Project-Based Learning

1 Introduction

Trends in Information Technology (IT) industry call for substantial changes in computer science education. After the decrease of dot-com-businesses, higher competition among graduates has reached computer science, too. In this situation, it is necessary that computer science students will be well prepared for entering the workforce. Future employers ask for real-world experience of their potential employees. In addition to factual knowledge and skills in the traditional curriculum, "soft skills" such as project management, communication and social competencies become more and more important. Nevertheless, they are still underrepresented

Manuscript received February 9, 2005
The authors are with University of Bremen, Department for Computer Science, Am Fallturm 1, 28359 Bremen (e-mail: abreiter@informatik.uni-bremen.de).

in computer science and software engineering curricula. Project-based learning is regarded as a suitable approach for more authentic, student-centered learning to acquire these skills. There are already documented efforts and results of project-based learning in engineering, and increasingly in computer science, which will be reflected later. Especially for the first phases of software development more social than technical skills are necessary. Requirements analysis is regarded as a key activity that includes eliciting, structuring, and representing clients' and users' expectations. Hence, there is a need for social and communicative skills, which are traditionally not well developed by software system engineers, nor are they integral part of computer science education.

At our Faculty of Mathematics and Computer Science at the University of Bremen, students have to participate in a four-semester team project that ideally follows the software development cycle from requirements analysis and definition to specification, coding, implementing, and usability testing. Student team projects are an integral part of the curriculum. We assume that software engineering needs social skills that can best be learned in authentic, student-centered and team-oriented projects. The paper describes the general curricular concept of project-based learning in student teams. In two examples, one from applied computer science and one from technical computer science, the goals and objectives as well as the organization will be described. The focus of this paper is on the learning process for project students, not on the software product they have developed.

2 Necessary Skills for Computer Science Graduates

According to studies and job ads, employers in IT industry are seeking computer science graduates who possess high order thinking skills and communicative abilities. Each year, the National Association of Colleges and Employers (NACE) conducts a "Job outlook survey" to determine what qualifications employers consider most important in applicants seeking employment [1]. In 2003, the top three factors were (on a scale from 1 to 5): 1. Communication skills (4.7 average); 2. Honesty/integrity (4.7); 3. Teamwork skills (4.6). We can assume that many approaches to systems development expect communicative skills and, additionally, the use of ethnographic methods from system developers. Such as ethnographers try to observe groups, communities or individuals, software developers need to be specifically aware of the social and organizational context in which their system might be used. Participatory design was originally developed in Scandinavia to balance the interests of workers (unions) and management in software implementation projects [2], [3]. Qualitative methods are still in their infancy in software engineering – good examples and methods from social sciences are adopted slowly

[4]. In order to move from abstract user requirements to more concrete system specification, prototypes and extreme programming have also been investigated in software engineering over the last decade [5], [6], [7]. Both can be regarded as advantageous compared to other software development models when there is a need in finding missing, previously unknown requirements during development.

On the other hand, adjusting system specifications continuously to randomly changing interests (tastes) of users could cause problems and slow down the development process, which requires communicative skills on both sides. Additionally, it cannot be taken for granted that users can always state explicitly their tasks and workflows. It is important to have in mind language barriers between systems designers and end-users. The more the requirements analysis is using example tasks (scenarios) from real life and small models of the future product (prototypes), the more the product fits into user's workflows and meets their expectations (see fig. 1).

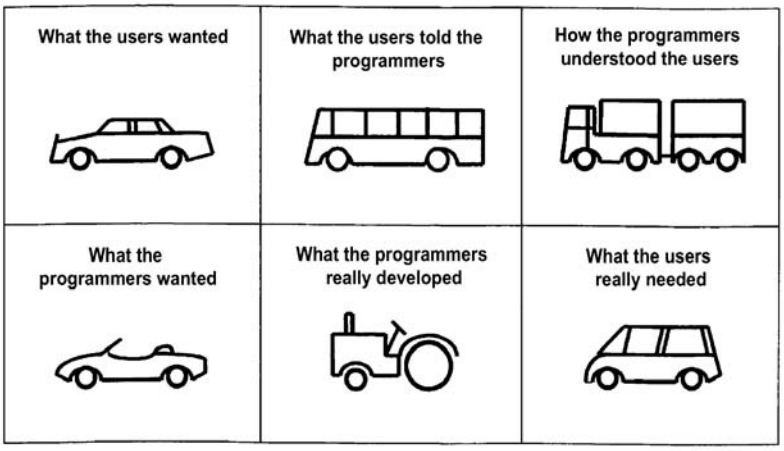

Fig. 1. Adapted from [8], p. 21

In addition to concrete technical knowledge, the curricula have to be extended with these contents. In 1998, the "Computer Science and Software Engineering Education Relevance Survey" identified that "Requirements Gathering & Analysis" is under-taught in university education [9], [10]. The study pointed out, that the expectations from future employers are much higher than the actual skills acquired at the university. Many scholars agree, that there is a growing need for innovative approaches to software engineering education [11], [12]. The 2001 Computing Curricula (CC2001) [13] reflect these principles as well as the "Strawman Draft of Computing Curricula 2004" [14]:

"CC2001 must include professional practice as an integral component of the

undergraduate curriculum. These practices encompass a wide range of activities including management, ethics and values, written and oral communication, working as part of a team, and remaining current in a rapidly changing discipline" [13].

"Today, the information systems specialist plays a key role in determining the requirements for an organization's information systems and is active in their specification, design, and implementation. As a result, such professionals require a sound understanding of organizational principles and practices so that they can serve as an effective bridge between the technical and management communities within an organization, enabling them to work in harmony to ensure that the organization has the information and the systems it needs to support its operations. [...] Such skill sets include interpersonal communication skills, team skills, and management skills as appropriate to the discipline. To have value, learning experiences must build such skills (not just convey that they are important) and teach skills that are transferable to new situations." [14]

With the changing environment for graduates, it is clear that students need both knowledge and skills to succeed. This need driven by workforce demands for high-performance employees who can plan strategically, collaborate and communicate in teams. Additionally, graduates need to learn social responsibility and understand their roles as responsible citizens in a globalized economy and society.

3 Project-based Learning in Computer Science Education

Project-based learning is a model that organizes learning around projects. According to the definitions found in handbooks, projects are complex tasks, based on challenging questions or problems, that involve students in design, problem-solving, decision making, or investigative activities; give students the opportunity to work relatively autonomously over extended periods of time; and culminate in realistic products or presentations. Other features found in the literature include authentic content, authentic assessment, teacher facilitation (coaching) but not direction, cooperative learning, and reflection (e.g. [15, 16, 17]).

"Doing projects" is a long-standing tradition in educational settings, incorporating "hands-on" activities, developing interdisciplinary themes, conducting field trips, and implementing laboratory investigations. Project-based learning has strong ties to this tradition. Additionally, it uses new insights from research in neuroscience and psychology. There, it has become clear that knowledge, thinking, doing, and the contexts for learning are inextricably tied. Learning is partly a social activity; it takes place within the context of culture, community, and past experiences. Constructivists believe that the most effective way for project students to acquire knowledge is to apply information or instruction to assessing and resolving

problems that are common to their experience. With each new application, learners are forced to either modify existing knowledge concepts or develop new ones. Hence, it is crucial that learning occurs in its actual settings that are relevant to today's real world problems and to the project students' lived experiences. This didactical concept is referred to as authentic learning. Authentic learning implies that learning is centered round authentic tasks and is guided with teacher scaffolding, that students are engaged in exploration and inquiry, and that they have opportunities for social discourse. Furthermore multiple resources should be available to students as they try to solve meaningful problems. Authentic learning is student-driven and therefore demands creativity and discovery in and outside the classroom. In constructivist learning models it is essential, that content knowledge is embedded in the situation in which it is used.

3.1 Examples for Project-based Learning

The first documented effort can be found in the engineering curriculum at the University of Aalborg (Denmark). Since 1974, project-based learning is the core of the curriculum. This kind of learning environment demands a high degree of supervision and space for the project students. Accordingly, the continuous feedback loops requires lectures to be changed and renewed. In Aalborg, experience and research has shown that the combination of problem-based learning is effective and it produces readily adaptable graduates with strong qualities in high-order thinking skills, such as problem solving, communication and generic technical knowledge [18]. Project-based learning is particularly strong in engineering. In 1995, the EPICS (Engineering Projects In Community Service) program at Purdue University (see http://epics.ecn.purdue.edu) was created as a team-based project for undergraduate engineering students [19]. Its aim is to create partnerships between student teams and local community organizations. In this joint project, engineering-based problems in the community should be solved. EPICS tries to include experience with design as a start-to-finish process by defining, designing, building, testing, deploying, and supporting real systems. Within the long-term projects that span several semesters, the idea is to bring engineering expertise to community organizations, which are affordable. More examples in project-based learning in engineering are documented in the "Guide for Problem Based Learning in Engineering" [20] with 12 case studies from the UK. The observed benefits are that students can apply their theoretical knowledge to practical situations, which helps them to better understand the theory and to develop new and powerful skills. According to the evaluation, the motivation of both students and staff is higher [20].

Some of these concepts are also adopted in computer science courses like EP-

CoS (Effective Projectwork in Computer Science) in the UK [21] or at the University of Hamburg [22]. Although there is a high demand for team project work, managing project work is difficult as computer science projects are:

expensive, demanding considerable supervision as well as technical resources;

complex, combining different aspects to satisfy various objectives ranging from technical skills to presentation skills;

continually demanding, set in the context of a rapidly changing technology which affects technical objectives and demands ever-evolving skills in both students and supervisors [23].

In summary, the academic community obviously regards team project work as a crucial component of any computer science degree. The discipline's professional societies emphasize projects and group work as an important preparation for professional practice. Project work, especially in groups, has many educational and social benefits, in particular involving students in active learning processes.

3.2 Student Team-projects for Participatory Design in a Computer Science Curriculum

At the University of Bremen, the academic year is divided into two semesters. One semester consists of 14 weeks of lectures, followed by twelve weeks for exams. We offer three types of courses (see fig. 2) : Diploma (9 semester), Bachelor (6 semester) and Master (4 semester).

Within the curricula, project-based learning has a long tradition. In the Diploma program, students chose a two-year team project in 3^{rd} and 4^{th} year after finishing the basic courses and a software engineering class. In the recently introduced Bachelor and Master program, the student team project (Bachelor or Master Project) is one year. In this paper, we will just refer to the Diploma project. It counts for 40 out of 270 credit points within the European Credit Transfer System (ECTS, every CP is worth 30 weekly work hours), and is usually followed by a diploma thesis (Diplomarbeit) which topic is often built on results of the team project.

In our case, we use project-based learning in a student group-project over two years. Similarly to EPICS, student team projects ideally follows the traditional software life cycle from requirements analysis, specification, design, to test, evaluation and support. The topics of the team projects differ every year. They range from programming a virtual soccer team to the development of geographical information systems or specific test tools for hardware design. Students select their projects according to their interest. This is supported by a project presentation day, which takes place six months before the actual start. In most student team projects, we are mainly interested in projects that are connected to workplace experience.

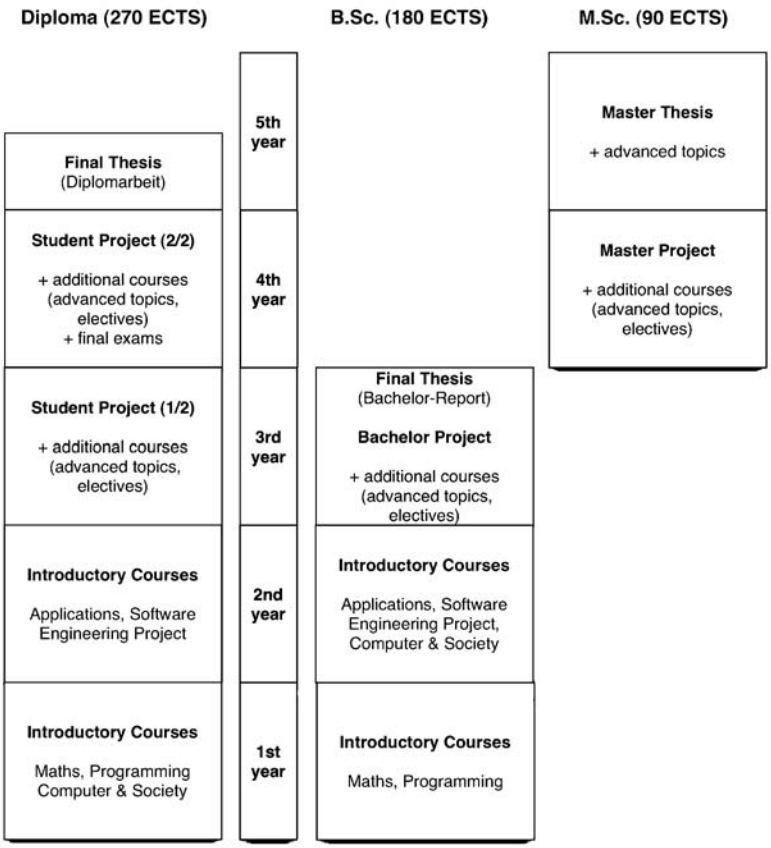

Fig. 2. Computer Science Curriculum Framework (Diploma, Bachelor and Master)

We will illustrate the concept according to two examples:

Project “learnweb”: Building a web-based system for collaborative work in schools

Project “FunTaskIC”: Building a design environment for integrated circuits

During two years, a group of approximately 20 project students is responsible for their own project management, and they have to produce a final report with all systems specifications, as well as implementing and testing the software system. The student teams get their own room and in some cases receive funding for equipment according to a business plan they have to develop during the first weeks of the first project semester. Project students exclusively work in teams. They organize and document their own work processes. Usually, the student team projects are subdivided in smaller teams, which perform subtasks (e.g. software modules).

Each project defines its own project management team and they take care of the work progress, time management, the plenary sessions that are held weekly and external relations to partners. The academic staff defines the guiding framework in which the project should take place. They work as coaches and facilitate the learning process. They provide input and help for specific topics. One day per week for four to six hours is reserved in the schedule just for the student project. We expect from students to invest at least the same amount of time outside the scheduled hours, running up to 15 hours of workload per week. Once a week, a plenary session is scheduled. They are used to discuss organizational tasks and to reflect the work process. Plenary sessions as well as lectures are regarded as working session, just as meetings are in industry. Additional accompanying courses (lectures and seminars) are offered to build the theoretical basis for the project students. Usually, project student present their work at plenary sessions, where they get feedback from course organizers. At the end of each term the individual performance of each student is evaluated. The quality of their presentations and work-in-progress count for their overall mark. After the two years, the student team projects present their results at a faculty-wide event where staff, students and public are present. The student team projects usually run once with the same topic but some are modular and the next generation of project students build upon previous work.

3.3 Example Project 1: Learnweb

Goals and objectives

The goal, which the project students developed together with the course organizers in the starting phase of the project, was to build a web-based groupware system for public schools. The final result of the project was a software tool (prototype).

Participants

The project team was composed of 21 students (7 female), 15 of them were from the same semester.

Similar to EPICS (see chapter 3.1), the approach was to provide community organizations with high quality but inexpensive software tools. We have selected as partners three schools (secondary, vocational and adult education) in a city school district with different histories in using technology for collaboration between schoolteachers and their students, between administrators and parents. The project and the participating schools signed a contract in which both parties agreed on the project goals, milestones and time budget.

Process

The course organizers offered accompanying seminars on learning theory, group-

ware, e-learning development strategies as well as usability and accessibility. The project students gave presentations, which were marked respectively. External researchers and developers were invited to exchange ideas. In cooperation with an external software company, the project students received help in professional tools for software development. A communication trainer introduced key concepts for teamwork and for project management. During the project, project students offered voluntarily tutorials to their fellow project students, e.g. about special issues in web programming or methods for code testing.

Our project-based approach allowed students to experiment with requirements analysis and offered new insights about a methodology for user participation. Developing a comprehensive picture of the system requirements within the organizational setting of schools, it was necessary to understand the objectives, tasks and processes of key stakeholders. In order to model workflows of client students, administrators and schoolteachers, classroom activities as well as after-school activities were observed. With the help of qualitative methods, such as semi-structured interviews and participant observations, schoolteachers' practices and collaborative activities were identified. The project students, divided in sub-groups of six to eight, worked closely together with their partner schools. They developed scenarios and defined interview guidelines [24], [25]. All requirements were summarized in a report (available in German language on request). The project students had to understand schoolteachers and client students' needs. Therefore a detailed task analysis was conducted by the project students, looking at (digital) tools, forms of information distribution between the different stakeholders and communication channels. The task analysis included questions about the amount of time invested and the problems occurred. The second step was to take a detailed look at all documents that are used in the school environment. The results were fixed in a requirements document, which was then negotiated with the end-users.

Product

After the phase of requirements analysis and definition, different prototypes were built and then discussed with end-users (see figure 3 for a screenshot of the portal page).

The software system, fully implemented in Java with Java Server pages (JSP) contains the following modules:

Personalized portal (MyLearning/MyTeaching)

Resource Management (for rooms, hardware, books etc.)

Communication Tools (E-Mail, bulletin boards, chat)

Group calendar

Document management (with version control, limited access)

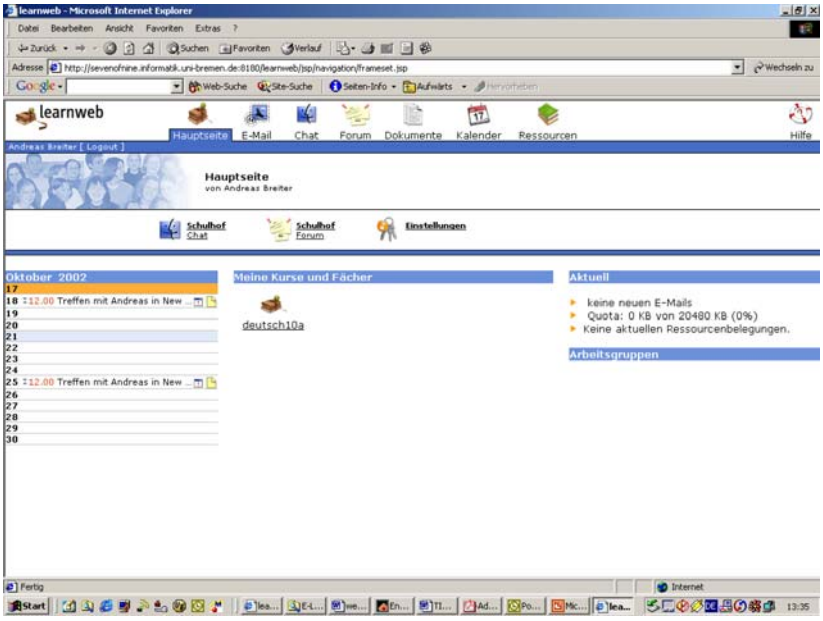

Fig. 3. Screenshot of the Groupware System (www.learnweb-bremen.de)

Project students conducted usability tests, which created more changes. These changes were included in the next prototype. After several iterations, a pre-final prototype was presented to the staff in all schools, including an intensive training for teachers. Following the contract, this was the final step of the project and the product was handed over to the schools, and continuous technical support was negotiated.

Results

There are obvious success indicators as employment after graduation or a tangible software product. Additionally, self-assessment of students' learning processes can play an important role. Continuous feedback by course directors and external project partners build additional empirical evidence for students' learning progress. Evaluating such a course is difficult as we follow a moving target. The course spreads over four semesters with many checkpoints, presentations and user inquiries. Hence, continuous feedback is necessary for the project students. During the two years of the course, every semester ended with individual assessments interviews about all aspects of the project (quality of code and documentation, design, management, marketing, team work, social skills etc.). The project students had the chance to explain their contribution to the overall project and the course organizers gave individual feedback. Additionally, project student performance was weighted

according to their individual presentations and research papers. At the end of the project, a quantum of 20 per cent of their final mark was negotiated within the team. Every project student graded her or his fellow project student respectively.

It has to be noted, that the project students who selected this project were specifically interested in acquiring "managerial skills". In all feedback conversations, they emphasized their interest in working as project managers rather than as programmers.

Positive	**Negative**
• Goal-oriented • Team spirit • Contact with end-users • Open learning environment (course organizers as moderators) • Peer-grading • Project management • Peer-tutoring ("best way of learning programming skills") • Acquired skills (both social and technical) valuable for future jobs	• Overambitious, especially conversations and tiring negotiations with schools • Gap between active and lazy project students • Very time and labor intensive • Less structured – more introductory lectures required • Difficult to judge fellow project students

Table 1. Results of project student interviews

A very strong and elaborated feedback came from the project partners, i.e. schoolteachers and project students (pupils). They gave an indirect rating of the project by using the product, indicating its usefulness. At the same time, schoolteachers appraised the performance of project student groups after presentations and trainings. They reported directly to project students and course organizers. In the progress of the project, the project students' abilities to present their own work and to discuss requirements and specifications with end-user increased significantly. The forms of communication within the team developed from chaotic meetings to

a professional work atmosphere with binding agreements, milestones and internal review phases.

3.4 Example Project 2: FunTaskIC

Goals and objectives

The overall goal was the development of an environment for the design of integrated circuits and systems. During the project the students with the help of the organizers further refined this goal and determined details like the design language that should be used.

Participants

The project started with 17 students (1 female), all of them from the same semester. While the project was running three participants left the project due to personal reasons. These students left the project after the first, within the second and after the fourth semester, respectively.

Process

Accompanying lectures and seminars were given throughout the project by the organizers. These courses covered the design flow for integrated circuits in general and also deepened particular stages of this flow. Additionally, in the first semester of the project the students themselves presented tutorials on different aspects ranging from professional development tools to techniques and algorithms for circuit design. This guaranteed that every participant had the basic technical knowledge for the project. Key issues for team projects were also discussed in tutorials prepared by the students. This knowledge was deepened for two students per project in a one-day workshop held by an external communication trainer.

During the second semester the specification of the final system was determined. This included the general discussion on the hardware description language to use, the software development language, but also the detailed analysis of the requirements for an integrated development environment. The Unified Modeling Language (UML) was used to aid the textual specification of the system. Milestones were discussed and set by the student team to enforce the product development. At this stage the team was split into five subgroups, one of them was responsible for project organization, the other four subgroups covered individual stages of the design process: entry, synthesis, verification and test. These groups had to interact tightly to specify common interfaces and file-formats, and to design the overall graphical user interface for the development environment.

Starting in the second semester the implementation of the individual stages was carried out by the subgroups. Within the third and fourth semester the different modules were assembled to form the complete system.

Product

So far there are only a few tools that support handling of the new hardware description language SystemC.

The final system contains modules corresponding to the different design stages:

Graphical design entry

Conversion to SystemC

Logic synthesis of SystemC code

Equivalence checking of the resulting logic representation

Test pattern generation for post-production correctness check

All modules can be controlled from a single graphical front-end (see Figure 4).

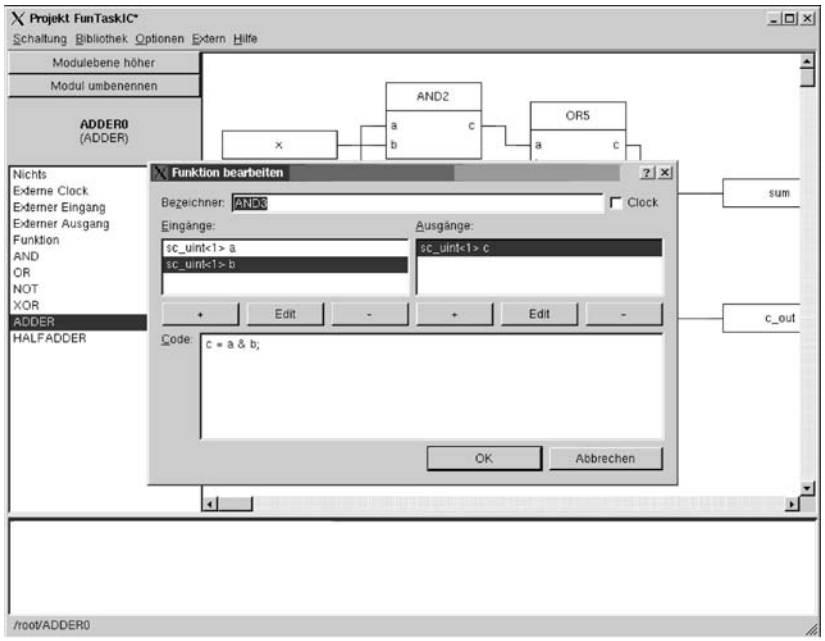

Fig. 4. Screenshot of the FunTaskIC design environment

Results

Interviews were regularly held with each student to evaluate the individual performance, but to reflect different aspects related to the project. These included – but were not limited to – the overall performance of the team, the relations and problems within the subgroup and between students, respectively.

For most participants the motivation to choose this project was the interest in computer architecture and to gather insight into the technical process of developing integrated circuits. These expectations were satisfied. But after the project the students also stressed the importance of the social experience. Some of the subgroups managed difficult situations. For example the leader of an industrial project is usually in a senior position compared to the project members. But the organizing group was confronted with the difficulty of having to manage a project without being in a superior position. Another difficulty for the whole team was that of members leaving the project. In that case the group had to compensate the loss of knowledge and working resources. Other problems were arising from misunderstandings in the communication, different levels of programming skills, diverging opinions about approaches for realizing ideas.

The final grades were given by the lecturers based on the individual students' effort. This was evident from regular reports, quality of tutorials and the participation in discussions as well as programming work.

4 Conclusions

In summary, our assumptions about learning effectiveness in respect to acquiring social and technical skills with student projects were positively supported and were in accordance to previous findings (see chapter 3.1). Although student team projects are more complex and costlier than other forms of teaching and learning as laid out in chapter 3.2, the outcomes are convincing. Most project students gained practical experience in fields that were supported by theory. Building their own project with authentic users, goals and objectives, using adequate methods, quality management and user involvement gave them the opportunity to self-directed authentic learning in a relatively ”safe” environment.

At least at our university, there is a growing interest from project students to understand the early stages of systems development and to develop necessary skills as well as to learn the required programming techniques. These are technical skills that could be taught in ordinary lectures. Even more important are the social competences learned in the project. This is a major advantage, as many project students will become rather software project managers than programmers - being responsible for future software developments. The traditional curriculum for computer science does not include any management courses, to teach the social competences necessary for leading a real-world software project. Moreover, we believe that learning-by-doing is the most suitable way to prepare the students for such job challenges.

References

[1] NACE, *Job Outlook Survey," National Association of Colleges and Employers*. NACE, 2003.

[2] A. Clement and P. v. d.] Besselaar, "A retrospective look at pd projects," *Communications of the ACM*, vol. 36, pp. 29–39, 1993.

[3] C. Floyd, W.-M. Mehl, F.-M. Reisin, G. Schmidt, and G. Wolf, "Out of scandinavia: Alternative approaches to software design and system development," *Human-Computer Interaction*, vol. 4, pp. 253–350, 1989.

[4] E. M. Trauth, "The choice of qualitative methods in is research," in *Qualitative Research in IS: Issues and Trends*, E. M. Trauth, Ed. Hershey, PA: IDEA, 2001, pp. 1–19.

[5] R. Budde, K. Kautz, K. Kuhlenkamp, and H. Züllighoven, *Prototyping: An Approach to Evolutionary System Development*. New York, NY: Springer, 1992.

[6] D. Schuler and A. Namioka, *Participatory Design. Principles and practices.* Hillsdale, NJ: Lawrence Erlbaum, 1993.

[7] K. Beck, *Extreme Programming Explained: Embrace Change*. Reading, MA: Addison-Wesley, 2000.

[8] J. Beer, "Systemspezifikation," in *Softwaretechnik. Praxiswissen für Software-Ingenieure*, P. Brössler and J. Siedersleben, Eds. München: Hanser, 2000, pp. 21–49.

[9] T. C. Lethbridge, "What knowledge is important to a software professional?" *IEEE Computer*, vol. 33, pp. 44–50, 2000.

[10] ——, "Priorities for the education and training of software engineers," *Journal of Systems and Software*, vol. 53, pp. 53–71, 2000.

[11] J. Z. Lavi, D. Dalcher, M. Mannion, and R. Gallant, "Engineering of computer-based systems - a proposed curriculum for a degree program at bachelor level," *IEEE Transactions on Education*, vol. 47, pp. 247– 253, 2004.

[12] F. C. Berry, P. S. DiPiazza, and S. L. Sauer, "The future of electrical and computer engineering education," *IEEE Transactions on Education*, vol. 46, pp. 467–476, 2003.

[13] ACM/IEEE, *Final Report of the Joint ACM/IEEE-CS Task Force on Computing Curricula 2001 for Computer Science*. ACM/IEEE, 2001.

[14] JTFCC, *Overview Report including a Guide to Undergraduate Degree Programs in Computing for undergraduate degree programs in Computer Engineering, Computer Science, Information Systems, Information Technology, Software Engineering. Strawman Draft*. Joint Task Force for Computing Curricula (JTFCC). A cooperative project of The Association for Computing (ACM), The Association for Information Systems (AIS), The Computer Society (IEEE-CS), June 1^{st} 2004.

[15] J. W. Thomas, J. R. Mergendoller, and A. Michaelson, *Project-based learning: A handbook for middle and high school teachers.* Novato, CA: The Buck Institute for Education, 1999.

[16] W. Diehl, T. Grobe, H. Lopez, and C. Cabral, *Project-based learning: A strategy for teaching and learning*. Boston, MA: Center for Youth Development and Education, Corporation for Business, Work, and Learning, 1999.

[17] F. M. Newmann and G. Wehlage, "Five standards of authentic instruction," *Education Leadership*, vol. 50, pp. 8–12, 1993.

[18] F. Kjersdam and S. Enemark, *The Aalborg Experiment. Project Innovation in University Education*. Aalborg: Aalborg University, 1994.

[19] L. Jamieson, W. C. Oakes, and E. J. Coyle, "Epics: Serving the community through engineering design projects," in *Learning to Serve: Promoting Civil Society Through Service Learning*, L. A. K. Simon, M. Kenny, K. Brabeck, and R. M. Lerner, Eds. Norwell, MA: Kluwer Academic Publishers, 2001.

[20] PBLE, *A Guide to Learning Engineering Through Projects*. Nottingham: University of Nottingham, 2003.

[21] P. K. Linos, S. Herman, and J. Lally, "Service-learning program for computer science and software engineering," in *Innovation and Technology In Computer Science Education*, ITiCSE Conference, Thessaloniki, 2003.

[22] M. Janneck and W.-G. Bleek, "Project-based learning with commsy," in *Conference on Computer Supported Cooperative Learning*, Boulder, CO, 2002.

[23] S. Fincher and M. Petre, "Project-based learning practices in computer science education," in *Frontiers in Education Conference*, Tempe, AZ, 1998.

[24] J. M. Carroll, *Scenario-Based Design: Envisioning Work and Technology in Systems Development*. New York: Wiley, 1995.

[25] K. Holtzblatt and S. Jones, "Contextual inquiry: A participatory technique for systems design," in *Participatory Design: Principles and Practices*, D. Schuler and A. Namioka, Eds. Hillsdale, NJ: Lawrence Erlbaum, 1993, pp. 177–210.