CDIO Projects in DTU's B.Eng. in IT Study Program

Sparsø, Jens; Bolander, Thomas; Fischer, Paul; Hansen, Thomas Kjærgård; Høgh, Stig; Nyborg, Mads; Probst, Christian W.; Todirica, Edward Alexandru

Published in:
Proceedings of the 7th International CDIO Conference, Technical University of Denmark, Copenhagen, June 20 - 23, 2011

Link to article, DOI:
10.4122/1.1000054694

Publication date:
2011

Document Version
Publisher's PDF, also known as Version of record

CDIO Projects in DTU's B.Eng. in IT Study Program

Jens Sparsø, Thomas Bolander, Paul Fischer, Thomas Kjærgård Hansen
Stig Høgh, Mads Nyborg, Christian W. Probst, Edward Todirica.

Dept. of Informatics and Mathematical Modelling
Technical University of Denmark,
DK-2800 Kgs. Lyngby, Denmark

ABSTRACT

Since the fall 2008 all B.Eng. study programs at the Technical University of Denmark have been based on the CDIO concept. The adoption of the CDIO standards and principles resulted in new or significantly revised study programs. As part of this effort design-build projects have been introduced on each of the first 4 semesters, and each semester-project spans several courses.

The aim of this paper is to describe the four CDIO semester projects in the B.Eng. in IT study, and – along with similar papers describing the other six B.Eng. programs – to provide documentation to accompany an exposition with stands providing additional information and with students demonstrating their projects. The paper is narrowly focused on the IT-study program.

At the time of writing this paper the students enrolled in 2008 have completed all four semesters in the new CDIO-based study plan, and the students enrolled in 2009 are currently in the process of finishing the 4th semester. Consequently, the paper is reporting on curriculum development which has been implemented, and for which experiences have gained.

KEYWORDS

Design-build projects, student-demonstrations, CDIO-based study programs.

1. INTRODUCTION

Since September 2008 seven of DTU's B.Eng. study programs have been based on the CDIO concept [1]. For most of the study programmes this change called for significant revisions of the study programs. Readers, who are interested in the considerations and the planning which went before this change, are referred to [2] that explain how we adopted to the CDIO-standards and adapted the CDIO syllabus, and that outline our plan of action for introducing the new CDIO-based study programs.

Two of our overall decisions related to the fulfilment of the CDIO-standards 3 and 4, and the partial fulfilment of standards 3, 6, 7 and 8 [3], were that each of the first four semesters must include a multi-disciplinary project, which relates to several courses on that semester.

Furthermore each study program must include at least two Design-Build projects, one on the first semester and one on the fourth semester or later.

The aim of this paper is to describe the four CDIO semester projects in the B.Eng. in IT study, and – along with similar papers describing the other six B.Eng. programs – to provide documentation which will accompany an exposition with stands providing additional information and with students demonstrating their projects.

At the time of writing this paper, the students enrolled in 2008 have completed all four semesters in the new CDIO-based study plan, and the students enrolled in 2009 are currently in the process of finishing their 4th semester. Consequently, the paper is reporting on curriculum development which has been implemented, and for which experiences have gained.

The paper is organized as follows. Section 2 gives an overview of the B.Eng. in IT study program and DTU's semester structure. Sections 3-6 present the four semester projects and experiences related to the specific projects. Section 7 discusses the lessons learned at a more general level, and finally Section 8 concludes the paper.

2. OVERVIEW OF THE B.ENG IN IT STUDY PROGRAM

All the B.Eng. study programs follow a common structure illustrated in Figure 1: On the first four semesters all courses are compulsory. With some variation across the different study programs this is followed by a semester with elective courses, a semester with industry internship, and a semester with the final B.Eng. project and a couple of additional elective courses. Each semester consists of a lecture period (13 weeks), an exam period (2 weeks) and a lab-period (3 weeks). The 3-week full-time lab-period can be an independent 5 ECTS course or it can be part of a larger 10-15 ECTS course.

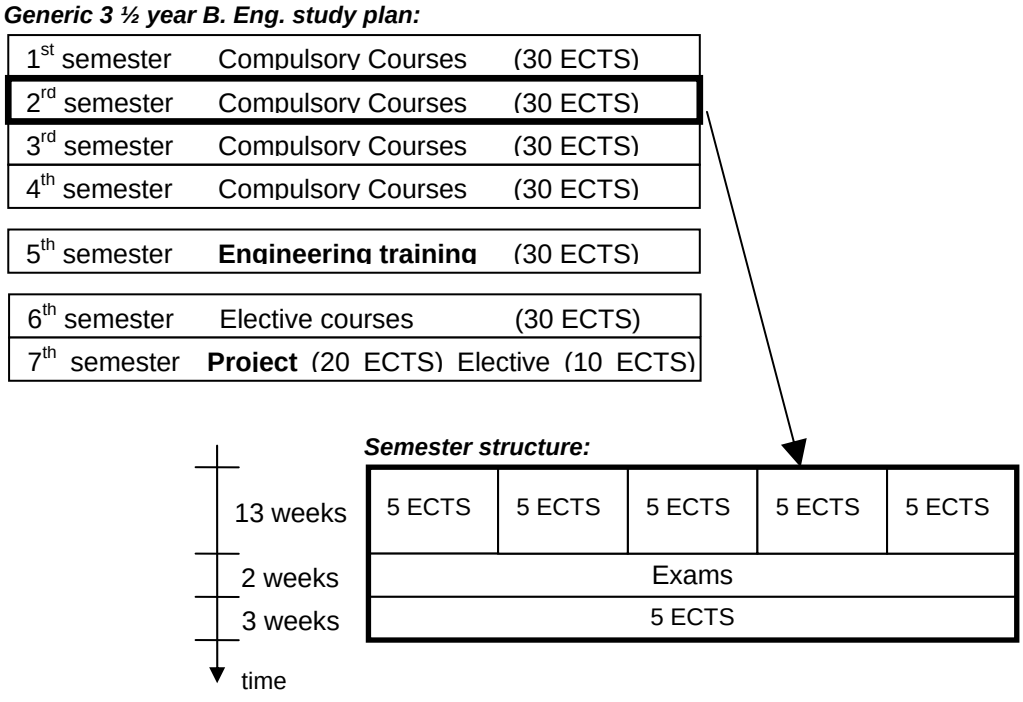

Figure 1. General structure of a 3 ½ year B. Eng. study.

	Semester period (13 weeks)					Lab period (3 weeks)
	5 ECTS	5 ECTS	5 ECTS	5 ECTS	5 ECTS	5 ECTS
1	01906 Math. 1	01917 Math. 2	31021 Electronics	02313 Dev. methods for IT systems	02312 Introductory programming	
2	01917 Discrete math. and databases	02311 Digital Systems	02325 Data communication	02326 Algorithms and data structures	02324 Advanced programming	
3	02323 Probability and statistics	02344 Objected oriented analysis and design (OOAD) and databases		02332 Compiler construction	02321 Hardware / Software Programming	
4	02333 Parallel and real-time systems		02341 Model-based software engineering	02342 Distributed systems	02343 CDIO project	
5	Elective courses					
6	Engineering practice / industry internship.					
7	Elective courses		Final B.Eng. project			

Figure 2. Study plan for the B.Eng. in IT study.

Figure 2 shows the specific study plan for the B.Eng.in IT education, and indicates which courses are involved in the CDIO semester projects.

The courses in the study program can be grouped as follows: (1) Mathematics and fundamentals (grey). (2) Programming and software engineering (green). (3) Embedded systems engineering (blue). The courses marked with a red frame cooperate on the CDIO projects. The 1st, 2nd, and 3rd semester projects are embedded in semester courses, which extend into the 3-week lab period. The 4th semester CDIO-project is a stand-alone course in order to emphasize that the project and project management is the crux of the matter.

The new study plan includes several changes besides the adoption of CDIO: (1) The amount of courses in *electronics and embedded systems* has been reduced and the amount of courses in *computer science and software engineering* has been increased. (2) As a result, several new courses have been introduced. (3) Finally, the sequence of courses was altered in a couple of areas in order to improve the semester-to-semester progression (see for example Section 2 in [4]). Because of this multitude of changes it is difficult to isolate and quantify the effect of introducing CDIO as discussed later in Section 7.

3. THE 1ST SEMESTER PROJECT – PROGRAMMING THE GAME MONOPOLY.

The 1st semester design-build project covers introductory programming, domain modelling and basic software development methods. Students implement a console-based version of the game Monopoly (Danish: Matador). The project starts with a sequence of weekly homework assignments focusing on different aspects of the overall software system. Towards the end these are integrated and the software system is completed and tested.

The advantage of developing a game is that it is popular among students. Furthermore, most students are familiar with the game from their childhood years. This has the advantage that the domain is well known, and at the same time rather challenging to formalize/model and

implement. All in all, the project contains many technical challenges related to the design of a robust object-oriented software system.
As shown in Figure 2, the project is supported by the courses *Introductory programming* (02312) and *Developing Methods for IT-systems* (02313). In course 02312 the technical disciplines are taught, and course 02313 coves the software development process. Students work in teams of about 4 students. Teams are set by the teachers at the start of the semester.

In the 13-week period the project is divided into 3 sub-tasks, which are intended to contribute items to the overall system. This also aims to train students' skills in writing reports and giving presentations to other students in the plenum, thereby training elements of the CDIO Syllabus 2 – 4. The 3 sub-tasks in the project are:

- Simple use of classes in Java and UML (Programming the dice).
- Use of classes and relationships (Programming the players).
- Polymorphism and domain modelling (Programming the fields of the board).

For each sub-task the groups hand in a project with an associated report. Each group presents their work to other students in a plenary session where they receive feedback on their work.

The project ends in 3-week period where the overall system is fully developed and integrated at the base of the 3 sub-tasks. As part of the final delivery, the students prepare a poster presenting the project. An example of such a poster is shown in Figure 3.

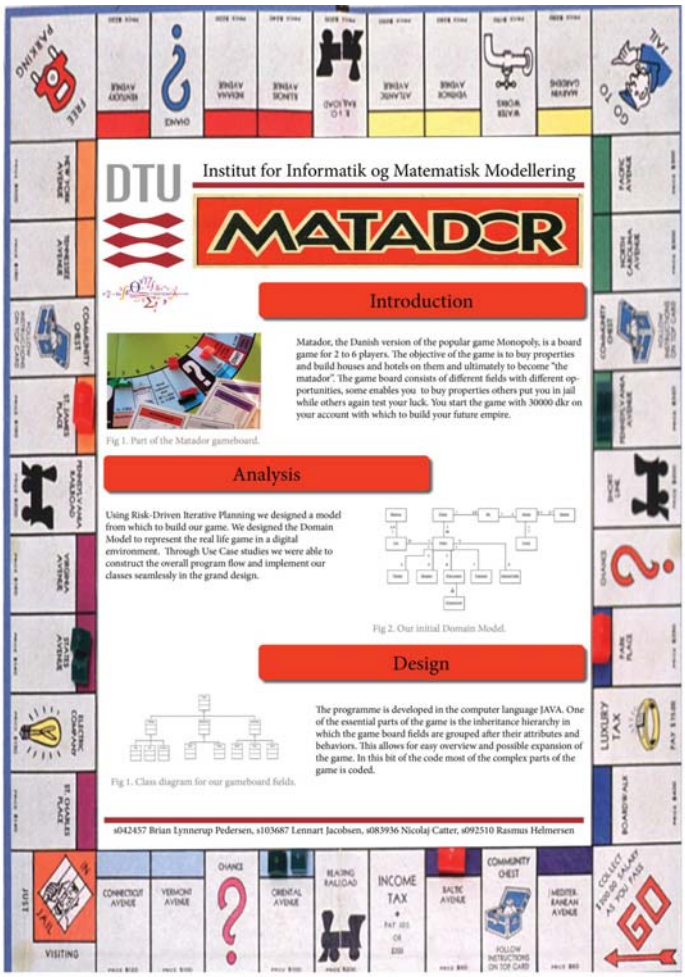

Figure 3. Poster produced by a team of students at the end of the course.

4. THE 2ND SEMESTER PROJECT – A DISTRIBUTED WEIGHING SYSTEM.

The 2nd semester project is again a programming and software engineering exercise. The project develops a distributed weighing system intended for use in a medical tablet production factory. The system, see Figure 4, involves a scale, a database server, and several computers implementing several user interfaces (an operator, foreman, pharmacist) all connected using a local area network. Further information beyond what is given below can be found in [5].

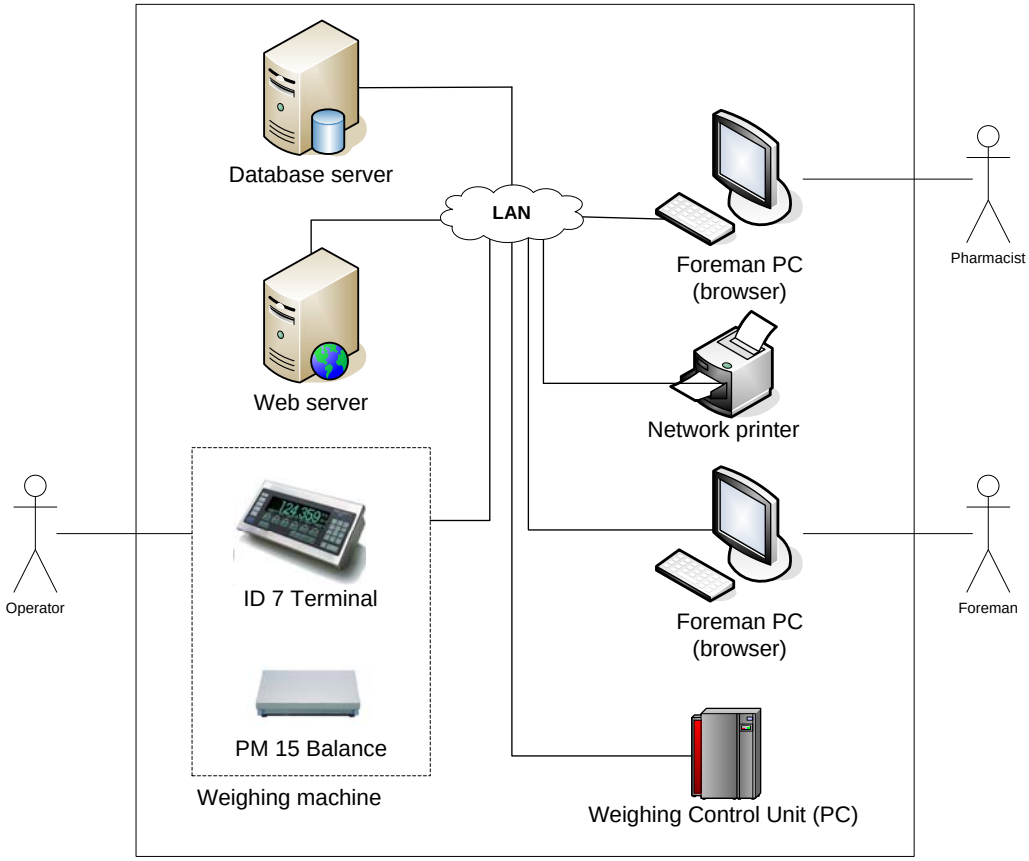

Figure 4. Context diagram of the distributed weighing system.

The project is embedded in the course *Advanced Programming* (02324), and as shown in Figure 2 the courses *Data Communication* (02325) and *Discrete Mathematics and Databases* (01917) are also involved – contributing to the project and drawing examples from it.

The students are divided into teams of approximately 6-8 persons formed by the teachers. The following criteria are used to form the teams:

- Students are not allowed in the same team as in the previous semester
- Students are placed in teams according to grades achieved in the previous semester (different grades in each team)

There are a total of 7 exercises during the 13-week period and a final report after the 3-week period. Each exercise is documented with a report. All 3 courses contribute to the 7 exercises in the 13-week period. Advanced Programming, which is the unifying course with the overall responsibility for the CDIO project, finishes the project in the 3-week period.

The purposes of the projects in the 13-week period are:

- To get a basic understanding of the overall domain

- To become familiar with development tools that are used in the project.
- To make the students accustomed to the operation of the weighing machine.
- To have students play their specific roles in the team.
- To get students to appoint a team leader

Key points that the students should consider in the final design are:

- User friendliness
- Security regarding production errors and general error registration
- Operator working speed when performing weighing procedure
- Traceability of production events and of final product back to ingredient suppliers
- System performance, including verification hereof by testing it with a generated full-scale dataset

It was a challenge to make the curriculum design fit into so many courses and the overall semester structure at DTU.

5. THE 3RD SEMESTER PROJECT – HW-SW CO-DESIGN / EMBEDDED SYSTEMS.

The 3rd semester design-build project develops a simple computer game using a combination of software and hardware. The project is open-ended in the sense that students can select which game they want to implement. The project is embedded in the course *Hardware-software Programming* (02321) as indicated in Figure 2. The course *Compiler Construction* (02332) relates to the project by developing a simplified Java compiler for the processor.

The overall aim of course 02321 and the associated project is to teach the fundamentals of how a processor and an embedded computer system work, both from hardware and from a software point of view. During the semester the course teaches: (1) micro-architecture and hardware implementation of a simple processor (4 weeks), (2) low level programming using machine code and assembly (4 weeks), and (3) high-level programming in C (4 weeks). During the subsequent 3-week period students design and implement small embedded computer system and implement a small application using it – typically some form of graphics based game.

Figure 5 shows a block diagram of the hardware platform, which the students implement on a FPGA [6]. The core LC-3 based computer system, along with the necessary software tools

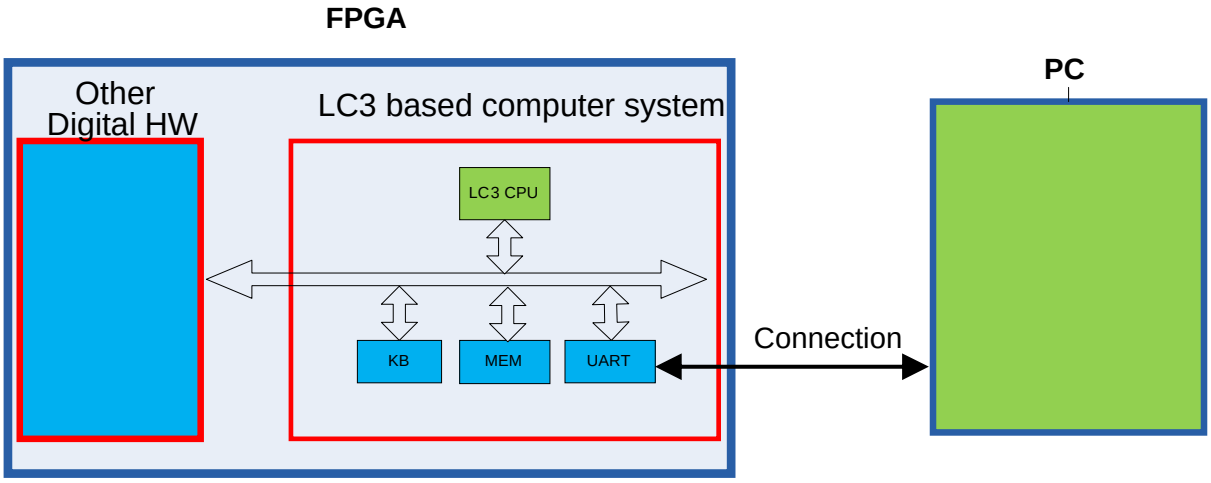

Figure 5. LC3 based computer system.

(assembler, simulator etc.), are introduced during the semester. For the project, students are provided with VHDL-code for the LC-3 processor (developed for this course) and VHDL code for the other hardware blocks are drawn from [7]. The task of the students is to integrate the components needed, reflecting a typical component-based approach to hardware design. The block called "other hardware" typically comprises a simple VGA display driver, and controllers for different types of input (keyboard, mouse, etc.).

The project specification is very open: "*To design and build a small computer-based system using the LC-3 processor and program an application for it".* Typical examples are graphics-based arcade-style games like packman, pong, tetris, hangman, space invaders, black jack, breakout etc. Figure 6 shows two examples: a space-invader type of game (left) and a form of racecar game (right). The software for these two games executes entirely on the LC-3. Other teams decide to use the PC for some of the functionality.

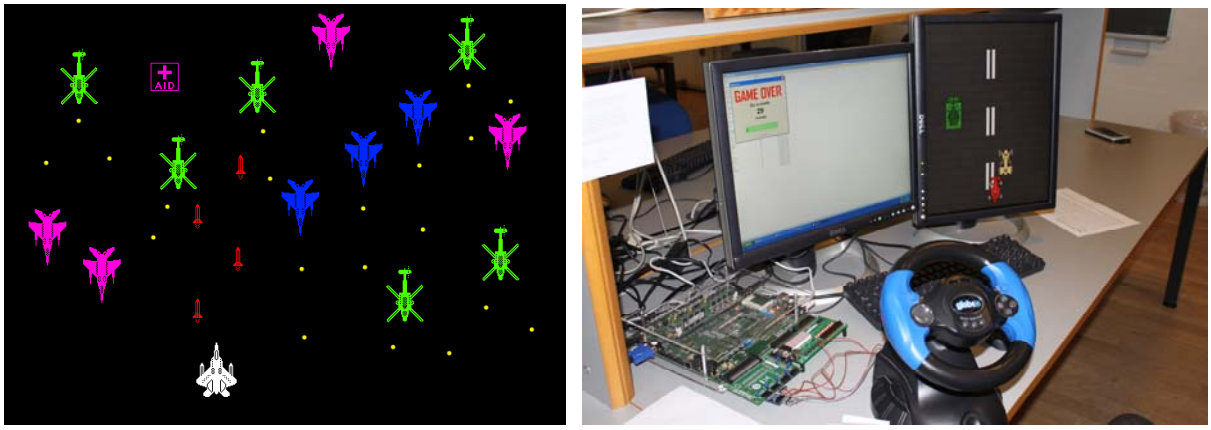

Figure 6. The most popular games from the last two edition of the course. A space-invader type of game (left) and a form of race-car game (right).

Although the project is closely linked to a single course, many students include topics from other courses in their project. Some groups have used the knowledge from course 02344 "Objected oriented analysis and design (OOAD) and databases" in order to store information about the game (for example high scores) into a database server running on the PC. Some groups have chosen to make statistics for some of the data collected from the game using knowledge gained in course 02323 "Probability and statistics".

In the course 02332 *Compiler Construction*, the students implement their own Java compiler for the LC-3 processor, and have the option of using it instead of the provided C compiler. In this course students design extensions to a base compiler [9], and can thus add project-specific support to the compiler used—from the level of the programming language to code generation for the LC-3 processor. While only supporting a restricted version of Java, the input programs can be compiled by standard Java compilers, a fact that has proven essential for student understanding throughout the project.

The courses and the project have been taught two times by now, and our experience is that the student interest in the courses is much higher than before the new CDIO-based study plan was introduced. Also, despite the fact that the project is rather complex, all teams manage to have a functional implementation that they can demonstrate at the end of the course.

6. THE 4TH SEMESTER PROJECT – A STAND-ALONE CDIO PROJECT.

The 4th semester project is a full-blown and stand-alone (10 ECTS) CDIO-project covering all 4 phases (conceive, design, implement and operate) of an engineering project. Students work in rather large teams on an open project and the course includes teaching of project management techniques. The course represents the conclusion of the first 4 semesters, and the conclusion of the compulsory part of the studyies. Information beyond what is given below may be found in [10].

Our key aims when developing the course were to:

- Cover all four main components of the CDIO model at comparable levels rather than putting exclusive focus on the design and implement phases.
- Provide the students with a challenging, ambitious, and exciting product development task inspiring creativity, originality and independent work.
- Achieve specific learning outcomes difficult or impossible to achieve in a standard lecture-based course: abilities to successfully apply previously acquired knowledge and skills in novel contexts, independently research and acquire new knowledge, achieve project management skills, achieve personal skills in presenting and representing both project and deliverables.
- Apply, combine and integrate curriculum elements from most of the mandatory courses.
- Achieve a high degree of efficiency in terms of the overall ratio between the obtained learning outcomes and the teacher/student resources invested.
- Achieve a reusable course model.

To achieve these aims a CDIO project was designed in which the students have to conceive, design, implement and operate a robotic multi-agent system from scratch integrating physical robot design, wireless communication, communication protocols, signal processing, image analysis, real-time systems engineering, software engineering, motion planning and path finding. The accompanying project description is very brief:

> *"You are given the necessary parts to design and build two Lego NXT robots, a web camera, building blocks for constructing mazes and a number of boxes. Use these components to build a robotic system that can remove arbitrarily placed boxes from within arbitrarily constructed mazes."*

Except for a few geometrical constraints on the mazes this is the entire project description, and it constitutes the only hard requirement the students have to meet. Other key features of the course are:

- Organization of the students in relatively large groups (5-7) encouraged to cooperate/compete at the same time.
- Inclusion of an end-to-end management process adapted from common methods in industry, including regular steering committee meetings and accompanying status reports.
- Lectures in project management supported by our own hands-on material. On the other hand there are no lectures on problem-domain topics, and no specific solution methods, techniques or technologies are provided.
- A final competition where the systems built by the different groups competed against each other in real-time, see Figures 7 and 8.

Figure 7. Students preparing for the final competition.

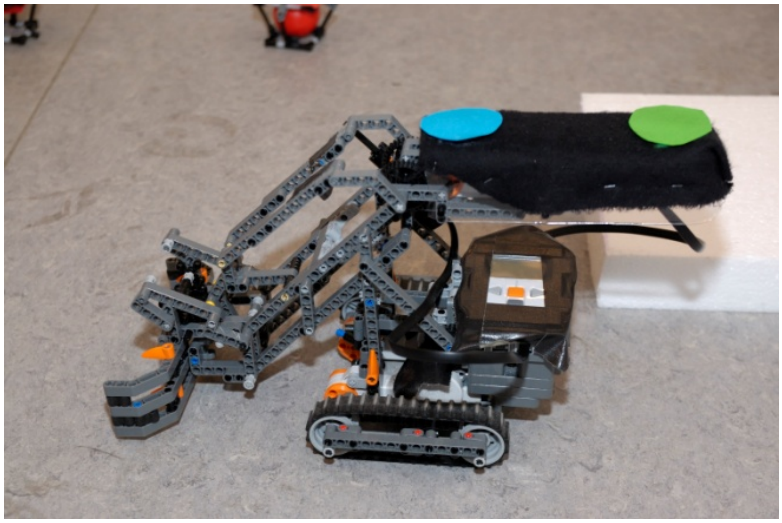

Figure 8. One of the custom robot designs used in the competition. The blue and green marks allow the web-camera to track the position of the robot.

The very brief and open problem description ensure that the conceive phase becomes a significant part of the design-build experience. The competition at the end ensures that the same is the case for the operate phase. The fact that no specific solution methods are provided forces the students to do independent research in and assessment of different possible technologies and methods; both among those methods previously learned during their studies and among new methods discovered through independent literature studies.

In the first part of the course many students go through a phase of frustration, as they are not used to such open problem statements and such high demands on their independence. However, through effective project management all groups produce working solutions in the end, turning frustration into a success experience and a strong sense of accomplishment. Along the road from frustration to success all the required learning objectives are met, in particular the students become much more independent and much better at finding and integrating methods and solutions from different sources.

The result metrics from the first conducted implementation of the course were above average for non-CDIO type courses at the university, both in terms of student learning, completion rate and satisfaction. All students passed the course, and it received the best student evaluation among all courses in the diploma IT degree program. Time invested by students as well as by faculty and material resources provided was within the norm.

7. DISCUSSION AND LESSONS LEARNED.

The new study plan implements several changes besides the adoption of CDIO: (1) The amount of courses in *electronics and embedded systems* was reduced and the amount of courses in *computer science and software engineering* was increased. (2) The latter called for the inclusion of several new courses. (3) The sequence of courses was altered in a couple of areas in order to improve semester-to-semester progression (see for example Section 2 in [4]). (4) Last but not least, a process of change is often accompanied by significant engagement and enthusiasm, which in itself has significant positive effect.

At the same time (in 2007 and in response to the Bologna Treaty) DTU introduced a new 7-step *absolute* grading scale. As part of this, learning outcomes were specified for each individual course. Although this was initiated independently of the CDIO adoption, the process contributed towards fulfilment of CDIO standard 2, and typically several learning objectives – in particular those specified for the courses containing the semester-projects – deal with personal, interpersonal, and product and system building skills.

Because of this multitude of changes, it is difficult to isolate and quantify the effect of introducing CDIO. Below we briefly discuss some of our experiences.

For the B.Eng. in IT study the 4 design-build semester projects replaced at total of 11 smaller and course-specific projects in the old study plan. The result is a better integration of the courses and that the student motivation – which is the key enabler for learning – is strengthened. On the teacher side a positive side-effect is a better knowledge of the content and mind-set of related courses and this again enables better teaching and linking to topics covered in related courses.

In our experience DTU's semester structure supports CDIO-style design-build activities very well: (1) The 13-week lecture period is well suited for Conceive-Design activities and the 3-week period is well suited for Implement-Operate activities. (2) Many software projects follow an iterative and incremental process, where each iteration covers all four C-D-I-O phases, and here the 3-week period is well suited for the final integration and test of subsystems developed earlier.

In the IT study program all semester projects are embedded in 10 ECTS courses extending into the 3-week period and this works very well.

When discussing study related issues with the students, they are generally aware of the fact that the study is based on the CDIO concept, and they indicate that they appreciate this. The fact that students in the 3rd semester – following their personal initiative – have included elements from other sources in their project is also very positive from a learning perspective.

8. CONCLUSION.

Since the fall 2008 all B.Eng. study programs at the Technical University of Denmark have been based on the CDIO concept. As part of this effort, design-build projects have been introduced on each of the first 4 semesters, and each design-build semester project spans

several courses. The paper presented the four semester projects in the B.Eng. in IT study and reported on the experiences gained from teaching these project courses. For the B.Eng. in IT study program perhaps the biggest change is that the four design-build semester-projects have replaced a total of 11 smaller and course-specific projects in the old study plan. The result of this is a much tighter integration of the courses and a general awareness of CDIO concepts among both faculty and students. This again has resulted in higher student motivation and higher student evaluations of the courses.

REFERENCES

[1] E. Crawley, J. Malmqvist, S. Östlund and D. Brodeur 2007: *Rethinking Engineering Education. The CDIO Approach.* Springer, New York 2007.

[2] J. Sparsø, P. Klit, M. May, G. Mohr, M.E. Vigild, "Towards CDIO-based B.Eng. studies at the Technical University of Denmark," *Proceedings 3rd International CDIO Conference*, 2007

[3] "The CDIO standards", Available at http://www.cdio.org/tools/cdio_standards.html (April 2007)

[4] M. May, L. Sendrup, J. Sparsø, T.K. Johansen, "E-learning support for student's understanding of electronics", *Proceedings of the 4th International CDIO Conference*, 2008.

[5] M. Nyborg, S. Høgh, "Mapping an industrial IT project to a 2nd semester design-build project", *Proceedings of the 6th International CDIO Conference*, Montreal, June 15-18, 2010.

[6] Yale N. Patt and Sanjay J. Patel, *Introduction to Computing Systems: From Bits and Gates to C and Beyond, 2/e.* McGraw Hill, 2004.

[7] Pong P. Chu, *FPGA Prototyping by VHDL examples – Xilinx Spartan-3 Version,* Wiley, 2008.

[8] Virtex-II Pro Development System, http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,400,794&Prod=XUPV2PNN

[9] Andrew W. Appel, *Modern Compiler Implementation in Java,* (Second Edition), Cambridge University Press, 2002.

[10] Thomas Bolander, Paul Fischer and Thomas Kjærgård Hansen, "From Frustration to Success: A Case-Study", *Proceedings 7th International CDIO Conference*, 2011.

Biographical Information

Jens Sparsø is professor in Computer Science and Engineering at the Department of Informatics and Mathematical Modelling at the Technical University of Denmark, and he is director of studies for the B.Eng. in IT study program. His current research activities are within application specific computing structures and processors, low power design techniques, design of asynchronous circuits and systems, and communication structures for systems-on-chip (i.e. Networks-on-Chip).

Thomas Bolander is associate professor in Computer Science at the Department of Informatics and Mathematical Modelling at the Technical University of Denmark. His research field is logic and artificial intelligence. Of special interest is the use of logic for modelling and emulating human-like planning, reasoning and problem solving. This is utilised in creating computer systems that can act autonomously and behave "intelligently".

Paul Fischer is associate professor in Computer Science at the Department of Informatics and Mathematical Modelling at the Technical University of Denmark. His research areas are algorithms, data structures and complexity theory.

Thomas Kjærgård Hansen is an External Lecturer at the Department of Informatics and Mathematical Modelling at the Technical University of Denmark. He holds a M.Sc. degree from the Technical University of Denmark and is currently operating his own engineering firm. At DTU he lectures on project management.

Stig Høgh is associate professor in software engineering at the Department of Informatics and Mathematical Modelling at the Technical University of Denmark. He has several years of experience in teaching in software engineering and has governed industrial projects both as consultant and as supervisor for student projects. He has in the period 1985-2005 produced software for quality control. This is sold in cooperation with the company Mettler Toledo Denmark.

Mads Nyborg is associate professor in software engineering at the Department of Informatics and Mathematical Modelling at the Technical University of Denmark. He has several years of experience in teaching in software engineering and has governed industrial projects both as consultant and as supervisor for student projects. He was the main responsible for introducing the CDIO concept at the diploma education at DTU informatics.

Christian W. Probst is an associate professor in the section for Language-Based Technologies at the Department of Informatics and Mathematical Modelling at the Technical University of Denmark. His research activities aim at realizing systems with guaranteed properties. An important aspect of his work relates to abstract machines for robustness and high-performance computing as well safety and security properties, most notably insider threats against IT systems and organizations.

Edward Todirica is teaching computer science and engineering courses at the Department of Informatics and Mathematical Modelling at the Technical University of Denmark. His areas of interest include processor design, embedded systems, digital design using FPGAs, hardware-software co-design.

Corresponding author
Jens Sparsø
Department of Informatics and Mathematical Modelling
Technical University of Denmark
Richard Petersens Plads
Building 322, room 215
2800 Lyngby
Denmark
Phone: (+45) 4525 3747
Email: jsp@imm.dtu.dk