MAPPING AN INDUSTRIAL IT PROJECT TO A 2ND SEMESTER DESIGN-BUILD PROJECT

Mads Nyborg, Stig Høgh

DTU Informatics, Technical University of Denmark, 2800 Kgs. Lyngby, Denmark

ABSTRACT

CDIO means bringing the engineer's daily live and working practice into the educational system. In our opinion this is best done by selecting an appropriate project from industry.

In this paper we describe how we have mapped an industrial IT project to a 2nd semester design-build project in the Diploma IT program at the Technical University of Denmark. The system in question is a weighing system operating in a LAN environment. The system is used in the medical industry for producing tablets. We present the design of a curriculum to support the development of major components of the weighing system. A simple teaching model for software engineering is presented which combines technical disciplines with disciplines from section 2-4 in the CDIO syllabus.
The implementation of a joint project involving several courses supports the CDIO perspective.

Already the traditional IT-diploma education for decades has included many of the essential features of the CDIO (for example, focus on teamwork, development of social skills, the open nature of design problems).

The specific project has previously been conducted on 5th Semester

The project has now been brought forward to the 2nd semester of study. A successful implementation at this level requires careful planning of activities through the semester. Principles of the CDIO have been of great help in this regard.

Finally we draw conclusions and give our recommendations based on those.

KEYWORDS

Software Engineering, Design-Build project, Curriculum design, Team work

INTRODUCTION AND BACKGROUND

At the Technical University of Denmark it has been decided to introduce CDIO in the Diploma IT program. The first 4 semesters each contain a CDIO project. This has an impact on students who started their studies in 2008. This paper deals with the 2nd semester CDIO design-build project. The project now runs for the second time. A project with a very similar technical content has been used since 2003 in the 5th semester. The project here was associated with an advanced networking course (10 ECTS). It has been a challenge to implement this now as a CDIO design-build project, located in the 2nd semester. The project involves 17.5 of 30 ECTS points in total for this semester. We discuss the project in general and the actions we have taken to implement this successfully at such an early stage in the program. A key focus has been to ensure that students will be familiar with development tools, with the understanding of normal use of the current hardware and with the formation of a well-integrated social group.

THE PROJECT

The actual project is a weighing system used in the medical industry for producing tablets. The weighing machine used is a Mettler precision electronic balance together with a terminal operated in a LAN environment, see figure 1. The system has three actors: Operators, a pharmacist and a foreman. The operators produce tablets in batches from recipes. Based on a recipe, the machine informs the operators which ingredients to use and the quantity. The operators take ingredients from the ingredients store, where they are stored in batches with a unique ingredient number. For each ingredient batch, the system records the supplier and the delivery date. Upon completion of a recipe, tablets are put in a pill bottle, assigned with unique batch number. The role of the pharmacist is to define recipes, and the role of the foreman is to distribute work to the operators on a daily basis.

Figure 1: The weighing machine

The system consists of several components, all of which require different engineering skills to be developed. These components can be divided into three areas: A weighing control unit that controls the operation of the weighing machine, a back-end component (database) that stores data about users, recipes, ingredient suppliers and production events, and a web component for administrative matters, i.e. definition of users, recipes, ingredients and concrete products.
The system context diagram is shown in figure 2.

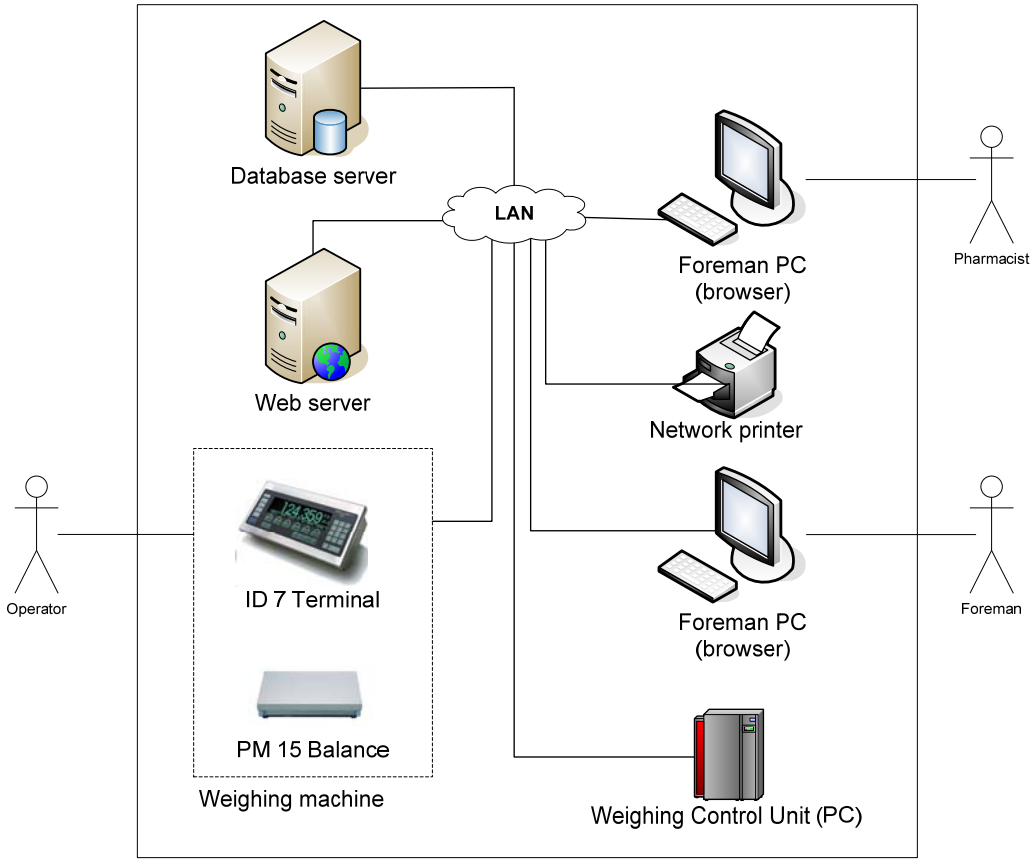

Figure 2: System context diagram

The realization of the system can be divided into a number of components each running on its own node. The overall architecture diagram for the system is shown in figure 3

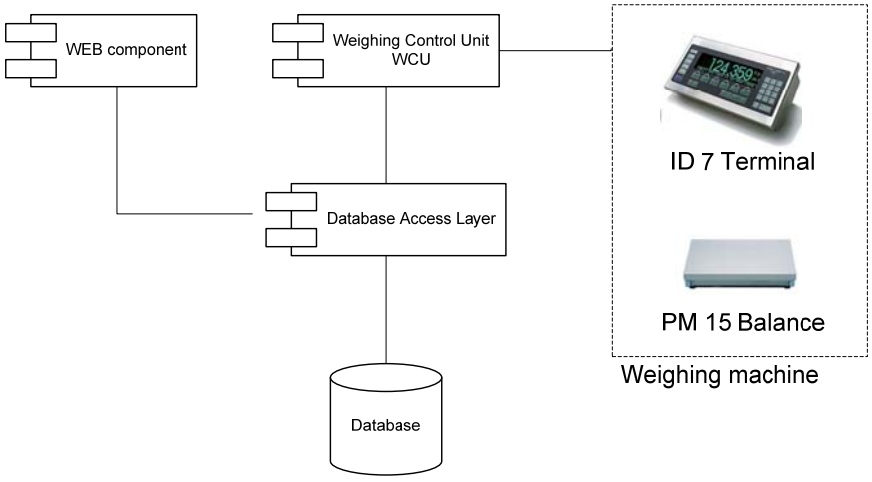

Figure 3: Software architecture

A key requirement for the system is the ability to trace all events during the production process. This comprises the registration of the operator who made the product, the recipe used and the identification of batches that ingredients were taken from. The reason for this procedure is the customer's requirements for full traceability in his production.

Quality control systems are often validated in the company where they are used. This may be the GMP (Good Manufacturing Practice) standard. This is a requirement in the pharmaceutical industry.
Validation can be implemented as a standard procedure with standard software. An example of this is Mettler FreeWeigh Net, [8], which is a standard validation module that can be purchased for implementation in the company. However, this is just an example, other methods and procedures are also used.

A SIMPLE TEACHNING MODEL FOR SOFTWARE ENGINEERING

Numerous books have been written on software engineering, e.g. [3]. but these are very comprehensive and in general not well-suited for teaching introductory courses.
Software engineering comprises all techniques in the construction of software systems. Overall these can be divided into technical disciplines and supporting activities.
We introduce a simple visual teaching model, see figure 5.

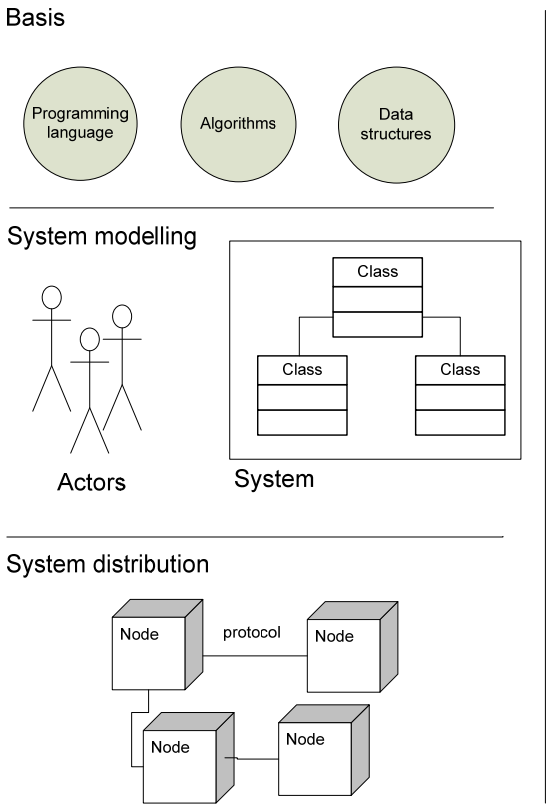

Figure 5: A teaching model for software engineering

The left part of the model shows core technical disciplines and the right part are supporting activities.

This model serves primarily as a simple one-page graphic overview of the following key parts:

Basis:
This part contains teaching in an object-oriented programming language (currently Java), algorithms and data structures and their application for solving a given programming problem, running on a CPU. The topics can be characterized as "classic computer science disciplines."

System modeling:
This part contains teaching in system modeling (currently object oriented system modeling), i.e. analysis of a domain and its application and building models for final mapping to a programming language

System distribution:
This part contains teaching of the issues arising from distributing a system to more than one CPU (so called nodes). This includes teaching in technologies, such as data communication protocols and frameworks used for implementation. Most IT systems today are distributed systems and the weighing system is a good example of this.

Supporting activities:
The supporting activities comprise all activities related to project work, e.g. teamwork, time planning, presentation etc. These are not directly taught as separate issues but integrated into the projects.
The model is introduced in the 1st semester and in order to make a basic understanding and acceptance of the key parts we start making comparison with a much more mature discipline, namely the building industry.
The basis part can be compared to the building elements, e.g. Brick, tile, tubes, doors and windows. However, knowing the function of these elements themselves is not enough to build a house.
To successfully build a house there is a need of an architectural drawing. The System modelling part of a software system is parallel to this.
The system distribution part can be compared to the challenge of building a city. In a city there are roads, junctions, roundabouts and rules for using them. This can be compared with data communication protocols in software systems. In addition, some of the buildings in the city have a special responsibility, such as post office, hospital, petrol station etc. This can be compared with the different responsibilities of the components in the weighing system.
The supporting activities are found in all engineering disciplines and hence also in software engineering. For software engineering this comprises most of the activities in the CDIO syllabus 2-4

CURRICULUM STRUCTURE

The Diploma IT program at the Technical University of Denmark consists of 7 semesters each of 30 ECTS point. All courses in the first 4 semesters are compulsory.
The semester structure consists of a 13-week period followed by a 3-week period. In the 13-week period courses are given in timeslots of 4 hours. Teachers may utilize these hour as they see fit, but hours are usually divided between short lectures, in-class problem-solving sessions, group exercises and project work. In the 3-week period the students work fulltime usually with some kind of lab projects.
Before the introduction of the CDIO concept, mini-projects were assigned to the courses and formed the basis of evaluation.
By the introduction of the CDIO concept we changed the curriculum structure to an integrated structure (standard 3). Several courses now contribute to a joint project.
The challenge was to make the curriculum design fit into the overall semester structure at DTU.
We use a variant of the merged curriculum structure ([1], p. 91) in which the 3-week period is the merging part where the students are supposed to finalize the project. The courses involved in the 13-week period are the following: Advanced Programming (02324), Data Communication (02325) and Discrete Mathematics and Databases for Diploma IT (only 2,5 ECTS of course 01917 comprising basic database technology are used directly in the project). Other courses in the semester which indirectly provide knowledge to the project are: Digital systems (02311) and Algorithms and Data Structures (02326).
From the 1st semester we utilize both technological and non technological skills (CDIO syllabus 2-4) obtained in the design-build project covered by Development methods for IT-Systems (02313) and Introductory Programming (02312).
The overall curriculum structure for the first two semesters is shown in table 1. (Note: courses in the 1st semester that do not serve as input to the 2nd Semester design-build project are omitted for simplicity)

Table 1
Curriculum structure for 1st and 2nd semester

	13 week period					3 week period
	5 ECTS	5 ECTS	5 ECTS	5 ECTS	5 ECTS	5 ECTS
1. Semester				02313 Development methods for IT-Systems	02312 Introductory Programming	
2. Semester	01917 Discrete Mathematics and Databases for Diploma IT	02311 Digital systems	02325 Data Communication	02326 Algorithms and Data Structures	02324 Advanced programming	

DEVELOPMENT PROCESS MODEL

The development model we use is based on the Rational Unified Process [2] and focuses on architecture and iterative development. Emphasis is put on team development, to strengthen students' social and interpersonal skills.

The students are divided into teams of approx. 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)

Each team appoints a project manager and a deputy, as applicable for the entire process. The supervisor is responsible for overall management of the team. We do not allow the students to form the teams themselves since this does not reflect normal working situations in industry.

Figure 4 illustrates how the projects are placed in the respective courses.

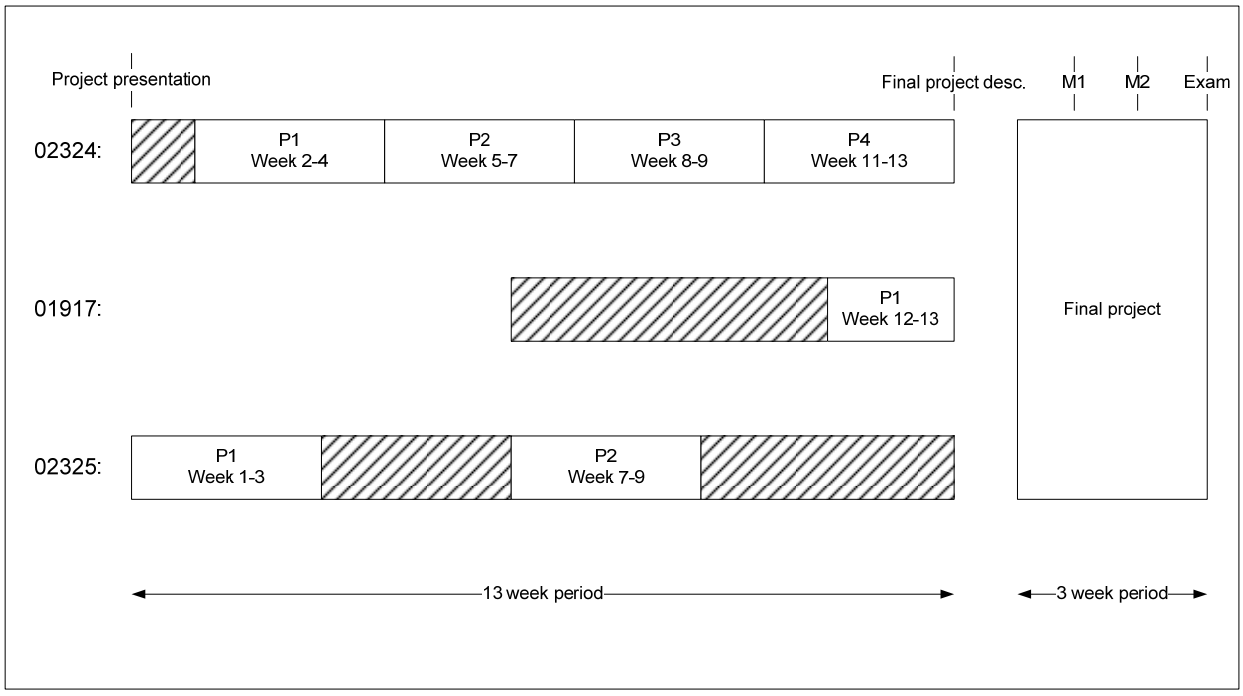

Figure 4: Timeline, projects and milestones

The overall project is presented in the beginning of the 13-week period. During the 13-week period, the participating courses define a number of small projects, all of which contribute to the development of the final system in the 3-week period.

The purpose of the projects in the 13-week period is:

- To get a basic understanding of the overall domain
- To get 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

Projects have a duration of 2-3 weeks. The project descriptions are limited to 1-3 pages. Most of the time spent in the lab will be related to the project.

The projects contribute to the understanding of the final system. The shaded areas mark periods in which other activities not directly associated with the project, take place.

The final project description (problem statement) is presented to students at the end of the 13-week period. This is a more comprehensive document which contains the requirements for the final system.

The projects in the 13 week period are described in table 2.

Table 2: Project description, background information and purpose

Course	P#	Project description / background information	Purpose
02324	1	Create a program that authenticates users based on username and password. The students are supplied with a proposed software architecture for the weighing system.	Get a basic understanding on how to work with layered software architecture.
	2	Create a program that simulates the operation of the physical weighing machine.	Get a detailed understanding of the weighing machine protocol.
	3	Create a web application that defines a simple user interface and validate input data according to a specified set of rules.	Get an understanding of basic web user interface design and validation procedures.
	4	Create a web application that authenticates users based on username and password.	Get familiar with a web authentication framework
01917	1	Create a program that is capable of retrieving and storing data in a relational database using embedded SQL. The students are supplied with the project data model, a database containing tables for the project.	Get a basic understanding of the SQL data manipulation language (DML) and how to use embedded SQL
02325	1	Create a standard communication program that is able to communicate with the weighing machine using a LAN connection.	Get a basic understanding on how to use socket programming and how the application protocol of the weighing machine operates.
	2	Create a program that implements a weighing procedure (Weighing Control Unit).	Get a basic understanding of the weighing procedure.

Design and implementation of the final system take place in the 3-week period.
The main objective during this period is for students to apply the general skills they have acquired in the 13-week period.
The introductory projects support the students' awareness of the functionality of subcomponents and to a lesser degree, how the entire system operates.

Key points which 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

Some considerations that support the above points could be:

- Entry procedures for miscellaneous identification numbers e.g. operator numbers, ingredient numbers, batch numbers, give the opportunity to design user-friendly procedures
- To verify that the correct components are selected in the weighing process
- General implementations of functions that handle error entries and incorrect operation
- The actual order of adding the ingredients in the weighing process: Is this 100% controlled by the system or is it possible for the operator to decide? When can the weighing procedure be interrupted?

DELIVERIES, MILESTONES AND FINAL EXAMINATION

Deliverables:
There are 7 exercises in the 13-week period and a final report after the 3-week period. Each task is documented with a report. Since it can be a challenging task to get software projects to work outside the development environment, we have specified exact procedures for delivery. It should be possible for teachers and examiners to import the projects on their own machines and make it run outside the development environment. Test data is implemented as well. In our experience quality is raised considerably and makes product evaluation possible. Usually you only have code and documentation available. It is motivating for the students and it puts an extra pressure that leads to a better completion of tasks.

Milestone presentations:
In the intensive part of the project (the 3-week period) there are 2 presentations (milestone M1 and M2, see figure 4). These presentations are role plays. In M1 the teachers play the role of the development manager in a software production company and in M2 the client who is buying the system. At times teaching assistants from courses may also be used. The teams have to present their project with focus on the current status.

Final examination:
The exam consists of three parts:

- Assessment of the final report and produced programs
- Oral presentation of final report and demonstration of the finished product followed by questions. All team members participate. The team determines who takes the floor during the presentation and who answers questions
- Finally, there is an individual oral examination of each student in the team. The student is asked question about the team's report and general issues within the curriculum

Grades are based on an overall assessment of these parts. Students are graded individually.

CONCLUSIONS AND FINAL REMARKS

It is our experience that the best teams are able to design systems at the same level as the ones being used in industry. The quality of the produced software is quite comparable to what is used in industry today. Within the business area of quality control using intelligent scales is a very beneficial solution.
Many companies, especially in the pharmaceutical industry, require that the systems are validated against different standards. This has not been a part of our project because it falls outside the focus of the syllabus.
We believe that the approach to introduce the 7 sub-assemblies is very important for a successful outcome.
It also provides knowledge of:

- The supplied software development tool. In practice this is a prerequisite for implementing the project
- The functionality of the weighing machine, especially how the software works in the operation of the weighing machine

Furthermore a good integration of the teams is obtained over a long process.

Having teams make presentation for each other may be considered. This will certainly be very useful for the students. Yet we have not selected this option. The argument for this is that we believe that it would be quite time consuming. In our experience weaker teams need help to prioritize between quality and quantity.

Overall the project described may seem to be very controlled. However, it is our experience that this is a necessity at this introductory level. Students at 2nd semester have little or no knowledge of architecture-matters of a distributed system. If we leave it up to the students to design for example the architectural model it would be time consuming and will also typically result in unmanageable systems. It is our experience that the few positive things that may be resulting from this is not in proportion to the time spent. Instead, we are trying to plan in such a way that students have the opportunity for a successful experience in terms of getting a complete system to work.

REFERENCES

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

[2] Kruchten, Philippe, The Rational Unified Process: An Introduction. Addison-Wesley , 2004 (3rd Ed.).

[3] Ian Sommerville, Software Engineering, Addison Wesley (7th Ed.)

[4] Sparsø, Jens ; Klit, Peder ; May, Michael ; Mohr, Gunnar ; Vigild, Martin Etchells, Towards CDIO-based B.Eng. studies at the Technical University of Denmatk, Proceedings of the 3rd International CDIO Conference

[5] Technical University of Denmark, Course description 02324: Advanced programming, http://www.kurser.dtu.dk/02324.aspx?menulanguage=en

[6] Technical University of Denmark, Course description 02325: Data Communication, http://www.kurser.dtu.dk/02325.aspx?menulanguage=en

[7] Technical University of Denmark, Course description 01917: Discrete Mathematics and Databases for Diplom IT, http://www.kurser.dtu.dk/01917.aspx?menulanguage=en

[8] Mettler Toledo, Validation Manual Volume 2, http://us.mt.com/global/en/home/supportive_content/product_documentation/product_brochures/Inhaltsverzeichnis_Validierungs-HB_2_FWN.rxHgAwXLuMvM.ExternalFileComponent.html/FWN_Validation_Manual2.pdf

Biographical Information

Mads Nyborg is associate professor in software engineering at DTU informatics. 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.

Stig Høgh is associate professor in software engineering at DTU informatics. 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.