A COURSE MODEL BASED ON LEARNING INSTEAD OF TEACHING

Bengt Lennartsson[1]

Abstract *—From a number of case studies in industry we have learned that the ability to learn new things is more important than the ability to remember facts and that the ability to question and to discover new trends is more important than the ability to follow detailed instructions. So we have tried to design a programming project course supporting the development of new skills or capabilities, but without loosing any of the traditional technical contents: distributed systems, computer networking, and client-server architectures together with object-oriented programming. We have used situations where the students have an initial understanding like modeling the facilities of a typical hotel. The student group develops a game-like system with a central information server keeping track of all the objects of the system and their states. The other subsystems are client programs modeling bars, escalators, gyms, gift-shops etc. Experience from different approaches during four years will be presented.*

Index Terms — Portfolio, software architecture, student directed learning, team learning.

BACKGROUND

A general trend all over the world is to widen the scope in engineering education. The traditional model of the development engineer is the ingenious inventor sitting at his desk or in the lab and understanding and solving the tricky technical problems himself. Today very few problems are such that they can be understood and solved by one person. Experience and knowledge from many fields will be required just for the problem definition, and the shared understanding needed in the team will evolve during the development process. Our findings, in short, from several longitudinal case studies [1]-[4] of the industrial development of complex systems, are that the main requirements on the members of the development team are:

- **Ability to learn new things** is more important than ability to remember old facts.
- **Ability to communicate and cooperate** is more important than individual brilliance.
- **Ability to understand totality and interdependencies** is more important than expertise in a narrow area.
- **Ability to question and to discover new trends** is more important than ability to follow detailed instructions.

Curricula mainly consisting of projects have successfully been used at Aalborg University in Denmark [5] since the early seventies. In particular in medicine the concept Problem Based Learning, PBL, [6] has been used mainly to motivate the student and support their 'learning how to learn'. In the PBL model it is normally assumed that the information needed is readily available in the textbooks and journals in the library. In engineering it is normally not so [7]. The development engineer must be able to solve *new* problems and develop *new* systems not described in the literature.

We have been using the three components of the old model of Aristotle [8] in our thinking about knowledge:

- ***episteme:*** the transferable declarative knowledge achieved by reading books and listening to lectures
- ***techne:*** the practical skills achieved by training and by understanding and copying behavior of the master
- ***fronesis:*** general wisdom, 'political knowledge', ability to interpret new situations and understand consequences of different actions in this new situation.

We believe that fronesis is very important today. In old Athens Aristotle considered *episteme* and *techne* to be sufficient for ordinary people. *Fronesis* was intended for the elite only, for those responsible for political decisions. Today, however, many strategic decisions have to be taken on the fly within the development projects. There is not time available to hand over the strategic decisions to some upper level.

The assessment of the students is simple and traditional as long as we focus on *episteme*: a written exam verifying that the student can repeat the material from the textbooks or the words from the teacher. The examination of *techne* is normally also straightforward: demonstrate ability to use tools, models and methods, to solve typical problems, or to design typical systems. The assessment of *fronesis* is much more complicated. Our approach is to try to use the portfolio concept [9]-[11].

COURSE LAYOUT

The course we will present here is a programming project course of four credits in the second year of an electronics design program. During the first year the students have had a course, five credits, on programming principles, where the basic concepts have been introduced. A second course, three credits, is on data structures and algorithms. Our course is the third in sequence and the last one in programming.

The intended technical content is software development in general and in particular computer networking, client-server architectures, and object oriented design and implementation. The programming language used is Java, and the platform is Windows NT. Most of the students have PCs of their own connected to the campus network. There

[1] Bengt Lennartsson, Linköping University, Campus Norrköping, SE-601 74 Norrköping, Sweden. *benle@itn.liu.se*

are around 70 students taking the course now given for the fourth time. The course runs over 16 weeks.

Objectives

Besides the intended technical content we are also aiming at project management issues. We leave it to the students to develop a general shared understanding of the problem and to define the system requirements. The idea is to give the students some experience from decomposing a total system into subsystems, from negotiating communication protocols and interfaces, and from organizing system integration and testing.

In addition to the objectives related to contents and result, we aim at a better use of the time for both students and teachers. At Swedish universities, in particular in engineering, it is still often considered necessary to cover all the contents of courses by lectures. The number of lecture hours is assumed to be related to the amount of material to be understood by the students. So another objective is organizational. We want to demonstrate to ourselves, to the students, and to our colleagues, that there may be better ways to organize a course and a better way to support the students in their learning.

Design Principles

In order to enable the students to define the system requirements themselves, we must choose a domain familiar to them such that they can understand and discuss the general needs and constraints. We have for four years used projects where the students have to develop game-like systems modeling familiar environments like an escalator system, a public transport system of a city, the facilities of a first class hotel, or a cruise ship.

In the systems there shall be a shared server function keeping track of the state of the system, and a number of client functions modeling facilities like casinos, discos, gyms, bars, etc. For each subsystem there is a group of six students responsible for the development. It is left to the whole group of around seventy students to define the client functions and, during the project, to define the interfaces and shared representations. At the end the system shall be operational on the Internet with the shared server function and the different clients.

It is a requirement that the design and implementation shall be fully object oriented. That is, it shall be possible, for instance, to instantiate different cruise ships with different discos, cafeterias, etc., on the fly.

In organizational terms we believe that the development of programming skills is based on a deeper understanding than just remembering phrases from textbooks or experience from trying different design solutions to some well-defined typical problems Such experience and deeper understanding must be based on interaction between action and reflection at the individual level. It does not occur synchronously among a large group of students in a lecture hall.

In particular in programming, we believe, the development of understanding and mental models takes place in a large number of small steps. During this process it is helpful for the students to have somebody to talk to and to exchange ideas and questions with. As many of the students in the course are in the same situation they will benefit a lot by talking to each other. So most of the problems and questions, far more than 90% of them, will be resolved in such discussions in small groups of students.

For the remaining problems, where the small group has discussed the problem without being able to find the solutions or answers, they will need teacher support. So the idea is, rather than broadcasting facts, concepts, problems, and solutions, to all the students two hours a week, to meet the students in small groups to resolve their particular problems where and when they are highly motivated.

Roughly, instead of spending two hours a week lecturing to the whole group, spend one tenth of that time on meeting each tenth of the whole group to give them what they really are asking for. In short, focus on optimizing the student's learning rather than on the teacher's lecturing.

Design Iterations

In the first version of the course we were mainly inspired by the experience from student projects at Aalborg University and the PBL approach at the Faculty of Health Sciences, Linköpings Universitet. The technical content was the modelling of a (complex) escalator system with different floor clients, cage clients, operator console clients, system creation clients, etc., plus a central resource allocation and state distribution server. For higher grades than pass, there were fifteen questions to answer individually for the next higher grade, and for the highest a special individual task, designed on request, was needed. There were a few preplanned lectures on topics where we assumed they would need some help. For higher grades than pass, there were fifteen questions to answer individually for the next higher grade, and for the highest a special individual task, designed on request, was needed.

The next year the course was slightly revised, but the basic structure remained. Both the first and the second year there were some problems with missing initial knowledge for some of the students, and the different levels of qualifications within the groups caused some problems

During the first three versions of our course we learned that there was a need to strengthen the initial shared conceptual platform in each group, so a specific pre-project has been designed to remedy this. We also learned that responding to the fifteen 'reflective questions' required for grade four was so helpful for both the students' and the teachers' understanding, that we wanted to make that a requirement also for 'pass'. Thus something new should be added as the requirement for grade four. The general structure for the course was, however, appreciated both by the students and by the teachers.

Current Design

In the current edition of the course the students have developed a cruise ship system, and we have defined the

requirements for grades three, four and five as presented below. This kind of grading based upon explicitly defined goals is common in the primary and secondary schools in Sweden, but it is very rare at the university level.

Requirement for grade three:

1. Before starting, present an individual declaration of initial understanding and ability in programming and a statement of personal goal in relation to the requirements for grades three to five
2. Contribute to the best of your ability, to the project work in general, to attend the weekly project meetings, and perform the tasks assigned to the deadlines given (such assignments and results delivered or achieved shall be recorded in the minutes of the weekly project meetings)
3. Give good support to the other students of the group in order to enable them to achieve their goals,
4. To have a good understanding, at the end of the course, of the meaning of the general concepts: class, object, constructor, class variable, class method, instance variable, instance method, client-server architecture, and the specific Java constructs interface, exception/throw/try/catch Vector, and to be able to design and implement systems built upon these general and Java specific components yourself.
5. Reflect upon what has happened during the project and deliver answers to the fifteen 'reflection questions'.
6. Participate and contribute at the final presentation of the work and convince the teacher that the requirements have been met, individually and by the group.

Requirements for grade four:

In addition to the requirements for grade three (please observe that items 2 and 3 are applicable also for students who have all the technological understanding from the very beginning)

To understand and be able to use and implement Threads, Animation, Serializeable, Runnable, Socket, and ServerSocket, and

To be able to use the components of AWT to design and implement appropriate graphical user interfaces.

Requirements for grade five:

In addition to the requirements for grades three and four, a deeper understanding of a particular area agreed upon between the student and the teacher.

Items 1 and 5 in the requirements for grade three are the first step towards a portfolio based assessment system.

The tradition in courses organized as projects is to regard a written report as the result of the project and then base the assessment of the students on the assessment of the report. This examination is often focusing on the group rather than on the individual students. In our case we regard the understanding and capability achieved by each student as the result for this student, and we have no project reports as such. The written answers to the questions of the reflection package are just an instrument to enable the student to reflect.

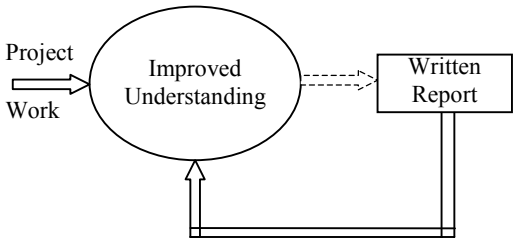

FIGURE. 1
THE MAIN RESULT IS THE UNDERSTANDING.
WRITING A REPORT MAY IMPROVE THE UNDERSTANDING.

The reflection Package

This reflection package is intended to support the reflection and self-evaluation done by each student during and after the project in the course. The package is based on the textbook used [12].

The reflection shall be done individually and be based on the notes you have taken continuously in parallel with your reading of the book. Selected questions from the reflection package:

1. In Chapter 1 of the book the importance of analyzing and understanding the problem before any design and implementation takes place. Is this in agreement with your experience from the project? Do you think you can acquire such knowledge from literature, or do you have to learn by your own mistakes?
2. How much of the time did you spend on analysis? Do you think it is possible to do such analysis without the understanding achieved from design and implementation?
6. Describe how your system and subsystems evolved. Which persons and which factors influenced the system structure? Did it change? If so, why? Mention the changes you would suggest if you had to restart now from the beginning?
7. How did you come to the conclusion about the functionality of your subsystem? How will it interact with other subsystems? How was this interaction described? Was there an initial understanding shared between the groups? How did this manifest itself? Would you like to propose any other way to establish an initial shared understanding?
8. How did you assign persons to the different task in your group? Was it based upon interest? Or previous knowledge? Or upon some other criteria? How do you think is should be done?
9. The members of your group may have different interests and priorities. Some were probably prepared to work very much, and somebody may aim at doing as little as possible. Do you think your student project is very different from projects in ´real industrial life´? How do you best manage such differences?
11. Chapters 9 and 10 are about the design and implementation of the code. How much of the time you spent on the project was coding? Do you think the coding fraction should have been greater? Smaller? List

the major difficulties you had in the coding!. What would you like to do differently in case you could start from the beginning today?

12. How much of the total time did you spend on test and integration? How did you plan these activities? Were test and integration more difficult than you expected? List the major difficulties! How could they have been avoided?
13. Chapter 14 is about estimating cost and time for a software project. Would it have been possible to use the methods presented in your project?
14. Was the total project suitably designed? List the major technical difficulties you didn't know of when you started! Anything that was much easier than you initially expected?
15. What have you learned by answering the questions in this reflection package?

The Portfolio Concept

The portfolio concept, originally used in arts for a collection of previous work, has evolved to a more general instrument for the assessment of progress in different areas.

The portfolio was introduced as an assessment instrument by the Harvard Project Zero [10] in the sixties. Then the concept has been widely used in primary and secondary school in many countries. Vavrus [9] defined portfolios as: '*a systematic and organized collection of evidence used by the teacher and student to monitor growth of the student´s knowledge, skills and attitude. The portfolio can provide an authentic and meaningful documentation of a student´s abilities, provided it contains the artifacts of the progress, as well as his/her reflections on both his/her learning and the artifacts chosen*'.

A Course Framework to Reuse

What we think can be copied and reused from our course is the following procedure:

1. Select an environment familiar to all the students, an environment with an appropriate number of interdependent facilities each suitable to modelling by a 'client program'.
2. a) Make sure that there are a number of groups of students, 5-7 students per group, and initiate brain storming sessions groupwise and with coordinating meetings with all students.
   b) In parallel, initiate design and implementation work in each group for a pre-project where the basic technical components of the real project will be needed. In our case socket communication, object streams to/from file storage and between Internet nodes, classes, objects, constructors, interfaces, graphical library components, etc. The idea behind this pre-project is to establish a basic shared technical understanding within each group.
3. Schedule weekly project meetings with the teacher for each group. Agenda for the weekly project meetings: verify that decisions at previous meetings have been acted upon, discuss and decide upon unresolved problems and questions, decide what shall be done until the next meeting and by whom. Agendas and minutes from all meetings should immediately be available from the web page of the group.
4. Enable the student group to decide about the total functionality of the system and the decomposition into server and client functions.
5. Let each student declare his/her initial understanding and programming skills and define expectations, ambitions, and a personal goal for the course.
6. Define dates for design review, subsystem test, integration test, and acceptance test. Assign at least one week after the acceptance test to reflection before the final examination.
7. At the end, give the groups and the students feedback on what they have done and on how they have done it. Assess the students by verifying that their goals have been met.

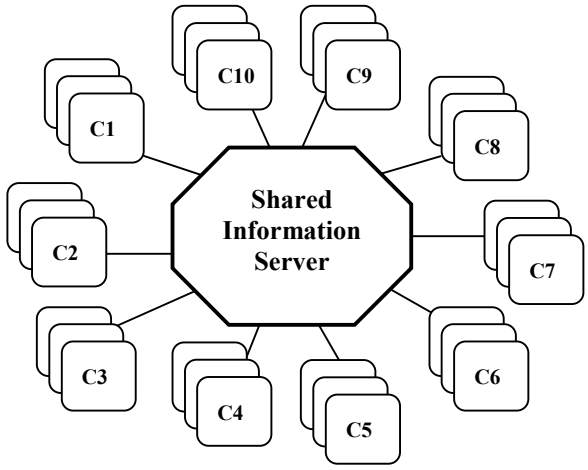

FIGURE. 2
GENERAL CLIENT-SERVER STRUCTURE OF SYSTEMS DEVELOPED.

This year we have had only one lecture, the introduction of the course structure and the objective. The technical contents have been available over the Internet. General text book material as Thinking in Java [13] and Java Tutorial [14] by means of links to the sources, and more specific examples and hints by means of web based material [15] as course bulletins are edited when needed. Assistance and advice from the lecturer and a teaching assistant have been almost continuously available via cell phone and e-mail.

Student Reactions

The more deep reactions from the students are explicitly available from their answers to the fifteen questions of the 'reflection package'. We have also used inquiries to get a statistical measure of the general attitudes to our approach. After the first year roughly one third of the students were very negative: '*it is the role of the teacher to plan the course and present the material to the students'*. Another third were very positive: *'this is the first time we have really learned something'*. Now when we have tuned the structure, the

information, and the assessment, we are still achieving the same goal, but the course is now highly appreciated by almost all the students.

As an illustration a reflection from one of the students follows:

When I started I didn't like having to spend many extra hours at the computer after the long period of programming project work, but now when I have reached the last one of the fifteen questions, I have started to appreciate this time-consuming reflection. I have spent time to sit down and think on what we have done and what we could have done differently. Thanks to this reflection task I have stopped and got a few new insights. Something I have noticed during this course is that you know a lot about yourself, but seldom you stop and think about it. If you don't give time to reflection and evaluation, you will not be able to learn and improve your behavior. I have now understood how important it is to establish a good communication between all parties involved in a project.

A quote from another student:

In this course, for the first time, I really feel that programming is funny! One thing that just occurred to me recently is that there was no regular teaching at all in this course! Everything is based upon guidance and general support. This is highly fascinating!

And a third voice:

I enjoyed the project. It was interesting and we got a clear picture of the framework and the general requirements already from the beginning. I think it a good idea to have a task that is funny and has some room for creativity and new ideas. I don't think a 'real' project aiming at the development of a 'real' system had inspired us students to work as hard and learn as much as we did in our project.

CONCLUSIONS

Our major conclusions from the four iterations of the course are:

1. Carefully designed tasks for small groups of students, together with regular short group meetings are a better model for the development of programming skills and for the understanding of system design and integration system than lecturing
2. Explicitly defined requirements for the different grades help the students in their goal setting.
3. Leaving the main responsibility for problem definition and for system decomposition to the students makes them more motivated and supports their understanding of the development of complex systems.
4. The portfolio approach to assessment looks promising, so far. When the students are responsible for the definition of their own goals they tend to be more ambitious from the beginning and more motivated and focused during the course.

PLANS FOR FUTURE

For the next year we intend to look at programming courses in the sequence and to apply the same learning model to all three. We will also elaborate on the portfolio assessment model. The general structure and the basic philosophy, as it has evolved, are likely to remain.

ACKNOWLEDGMENT

The approach to the university course is based on findings from industrial case studies funded by NUTEK, the Swedish National Board for Industrial and Technical Development.

REFERENCES

[1] Cederling, U, Ekinge, R, Lennartsson, B, Taxén, L, Wedlund, T: A Project Management Model based on Shared Understanding. *Proceedings of Management Minitrack in the Organizational Systems and Technology Track of the Thirty-third Hawaii International Conference on System Sciences (HICSS-33)*, January 4-7, 2000.

[2] Ekinge, R, Lennartsson, B, Taxén, L: Organizational Knowledge as a Basis for the Management of Development Projects. Presented at Discovering Connections: A Renaissance Through Systems Learning Conference. Dearborn, Michigan, September 24-27, 2000.

[3] Davidson, K, Lennartsson, B: Team Learning Capability - A Strategy to Master Complexity and to Achieve Flexibility. *Proceedings of The 7th World Conference on Continuing Engineering Education*, Torino, Italy, May 1998.

[4] Lars Taxén: Organizational Learning Revisited. *Proceedings of the First IEEE International Conference on Cognitive Informatics* (ICCI'02). August 19-20, 2002. Calgary, Alberta, Canada

[5] Kjersdam, F, Enemark, S: *The Aalborg Experiment - Project Innovation in University Education*. Aalborg University Press 1994. ISBN 87-7307-480-2.

[6] Albanese, M, A, Mitchell, S: Problem-based Learning: A Review of Literature on Its Outcomes and Implementation Issues. *Academic Medicine. Vol. 68. No 1. Jan. 1993*. pp. 52-81.

[7] Bengt Lennartsson: Why there are different teaching preferences in different disciplines. Proceedings of The International Conference on Project Work in University Studies, Roskilde, Denmark, September 14-17, 1997.

[8] Bengt Lennartsson, Evabritt Sundin: Fronesis - The Third Dimension of Knowledge, Learning, and Evaluation. *Proceedings of the 31st IEEE and ACEE joint Frontiers in Education Conference (FIE 2001)*, October 10-13, 2001. Reno, Nevada.

[9] Vavrus, L, G., and Collins, A. (1991). Portfolio documentation and assessment center exercises: A marriage made for teacher assessment. *Teacher Education Quarterly*, 3(2), 12-29.

[10] Perkins, D: *Smart Schools: Better Thinking and Learning for Every Child*. New York, 1992: The Free Press. ISBN: 0028740181

[11] Cole, Donna J, Ryan, Charles W, Kick, Fran*: Portfolios across the curriculum*. Corwin Pr, October 1999. ISBN: 0761975349

[12] Sven Eklund, Hans Fernlund: Prgram design with quality – project management and ISO-9000. (In Swedish). ISBN: 91-44-00626-8.

[13] Bruce Eckel: Thinking in Java. Paperback. 1128 pages. 2nd edition (June 5, 2000). Prentice Hall Computer Books. ISBN: 0130273635.

[14] Mary Campione, Kathy Walrath: The Java Tutorial – Third Edition, Alison Huml 2001. ISBN-0-201-70393-9.

[15] Course Information available from URL: http://www.itn.liu.se/~benle/about-tne007/