

# Project-Oriented Approach to Software Engineering Education in a Multidisciplinary Environment: Objectives, Realization, Evaluation

Silvije Jovalekic  
Information Technology and Engineering  
Fachhochschule Albstadt-Sigmaringen  
72458 Albstadt, Germany

## Abstract

*An essential part of Software Engineering Education is practical training in principles, methods and procedures under conditions similar to developing real software products. It helps the student to understand abstract principles in depth and to learn the art applying object-oriented methods. This paper describes Software Engineering Practical Training (SEPT) in the Laboratory for Microcomputing and Software Construction and the experience gained during the last four years.*

## Introduction

Well-founded education in Software Engineering for technical applications requires the knowledge of the principles, methods and procedures as well as their application in projects of modest size. Software projects should be done in teams and should include requirements specification, analysis, design, construction, quality assurance and project management. Project requirements have to be stated in such a way that the participants are confronted with the true problems of the specific applications. Because of the limited project duration and workload capacities the application problem should not be too complex. Final year undergraduate students in Information Technology and Engineering may choose the option Microcomputing Applications [1]. Software Engineering is one of the main areas in that option. It includes lectures in Software Construction, Real-Time Systems, Software Quality Assurance, Distributed Systems and multidisciplinary laboratory in Software Engineering.

## Training Objectives

Software Engineering Practical Training in a multidisciplinary environment pursues objectives of different categories: (1) it considers the fact that the problems and their software solutions are very diverse. Software development and supply presuppose machines of different types, e.g. PC, workstation, embedded microcontroller. Development of real-time software

differs very much from that of conventional one. The environment of a real-time system is simulated by process models, e.g. railway model. They are physical abstractions of technical processes and serve to model dynamic behaviour of the environment; (2) it recognizes similarities in diverse hardware/software platforms for software development. It makes use of basic software and standard tools available across platforms, e.g. make, yacc, rcs, real-time operating systems; (3) it emphasizes software development practice in teams. Different communication techniques are promoted: oral and written communication of ideas, computer cooperative work within a distributed environment. Effective use of computers are discussed and tried, e.g. rcs, email, sharedX, blackboards; (4) it stimulates goal-oriented work. Optimal solutions to the problems are developed and the software development process itself is optimized and adapted to the given problem. Standard development methods for analysis, design, implementation and documentation are applied, e.g. Object Modeling Technique, Gantt-diagram, C++ coding rules, documentation schemes.

## Training Overview

During their last semester the students have to complete two software projects. They are free to select from the given repertory. The aim of both projects is to practice object-oriented analysis, design and construction in general and real-time computing. The first project in general computing taking five weeks introduces the project methodology. The students may choose between the graphics class library and formal language construction. In the graphics project operations for graphical objects have to be designed, e.g. drawing, moving, rotating, group building, loading from and storing into files, etc. The language project involves declaration and manipulation of different variable types and simple control structures to interpret procedural programs similar to C.

The second project is done over eight weeks and considers real-time requirements of a given process model. Knowledge of electronics, mechanics and control

engineering is needed additionally to accomplish it successfully. A domestic heating automation system, a control system for a railway model or a real-time operating system may be selected. The idea for the heating project comes from [2]. The railway model project is discussed separately in detail. Participants interested in tool construction may develop the kernel of a real-time operating system consisting of time, task and interrupt management components [3].

Each project contains several small preparatory tasks and a complex problem. Preparatory tasks serve mainly to acquaints the participants with interfaces to the process model, with the software development environment and to narrow the gap between design and implementation.

Running different projects in parallel imposes additional supervision load. The main reason for having a variety of projects is to demonstrate the differences in software engineering problems and environments.

### Laboratory Equipment

Before making purchases of the current HW/SW equipment the projects were developed and partly tried with older equipment. The decision for the software tools was made first. The hardware was selected to meet the requirements of SEPT and the needs of the Computer Aided Electronics Engineering (CAEE) training programme which runs in parallel.

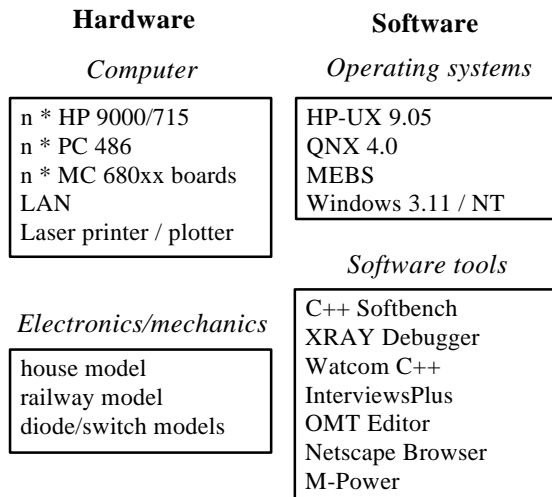


Figure 1. Hardware / Software Equipment

### Provided Documents

Training participants get a set of carefully prepared and selected documents needed to carry out the projects: descriptions of process models, descriptions of hardware abstractions and necessary operating system calls, problem statements, a Project Instruction Handbook and a

list of project documents to be delivered. The Project Instruction Handbook [4] consists of short descriptions and examples of requirements specification, design, implementation, project management, quality assurance und users manuals. Its aim is to assist the participants in all activities during the project. It is assumed that they already have a sound background in the described methods.

### Railway Model and Computing Facilities

The railway model consists of 27 track sections, 8 switches and any number of trains. Each section contains a sensor detecting the train passing it. They are direction insensitive. Identifiable trains can move bidirectionally with selected discrete velocity ranging from 0 to approx. 2m/s. The positions of the switches cannot be captured.

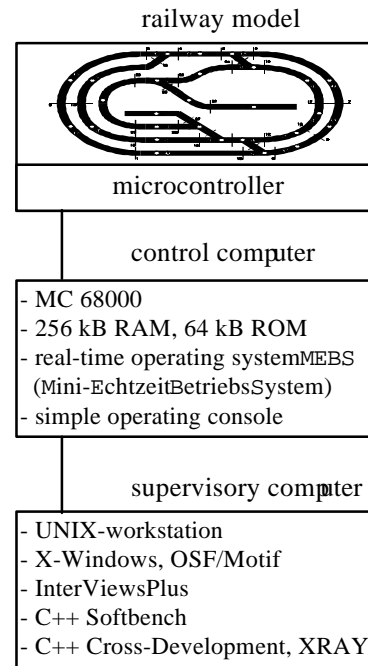


Figure 2. Railway Model with Computing Facilities

Figure 2 shows the railway model with computers at different levels. A microcontroller captures data from sensors and controls the trains and switches. It has to process signals within 250µs. It is connected to the control computer running real-time operating system MEBS and object-oriented real-time software. Table 1 shows various hard timing requirements which have to be met.

A simple operating console is interfaced to the control computer for human interactions to the railway model. A workstation connected to the control computer serves as a supervisory computer. The user can control

the switches, the trains and visualize their actual position by means of a user-friendly visual interface.

The students have to complete several small preparatory tasks: real-time programming using MEBS, identification of train position and siding of trains. The main problem consists of synchronization of several trains running at given speeds and different lines using same sections, design of the user-friendly human interface for interaction with the railway model and software to interconnect the supervisory and control levels of computers.

performed task	deadline [ms]
state capture	$\leq 50$
train access	$\geq 250$
switch access	$\geq 1000$

Table 1. Hard Timing Requirements

## Project Realisation

Projects are carried out in teams of 2-4 members. Each member has responsibilities similar to the development engineer in an industrial project. Additionally a special role is assigned to each member. Every team selects a project leader. He is responsible for planning project activities and milestones, software analysis and design process, system test plans and acceptance test. Other roles are quality and product managers. The product manager plans the version control for software products and provides the necessary support. The quality manager produces quality metrics data for object-oriented designs and checks if design and coding rules are followed.

## Supervision and Support

To fulfill the objectives within a given timetable active supervision is provided on a regular base. It consists of project meetings with supervisors and deliveries according to the supplied support plan. The team members are obliged to work in the laboratory for 4 hours per week. Additionally 4-6 hours are required to complete the project satisfactorily. Often they spend more time voluntarily and finish the project to a very high standard.

In week 1 the requirements specification has to be prepared. The team appoints the project leader, quality and product managers. The team and the supervisors clarify the requirements and classify them into necessary and nice to have. Special consideration is devoted to hard and soft timing requirements. In week 2 the responsibility matrix and the Gantt diagram for the whole project have

to be delivered. Preparatory tasks are performed in parallel with these activities.

In week 3 and 4 analysis and design documents are discussed. They consist of subsystem specifications, object and class models, task diagrams, class and task specifications and schedulability analysis. To describe analysis and design Object Modeling Technique (OMT) [5] and a graphical editor [6] are used. Figure 3 shows the class diagram for the railway model. The calculation of response time, processor utilization and estimating spare capacity is done using Rate Monotonic Analysis [7]. In week 5 the test plans for the whole system have to be completed.

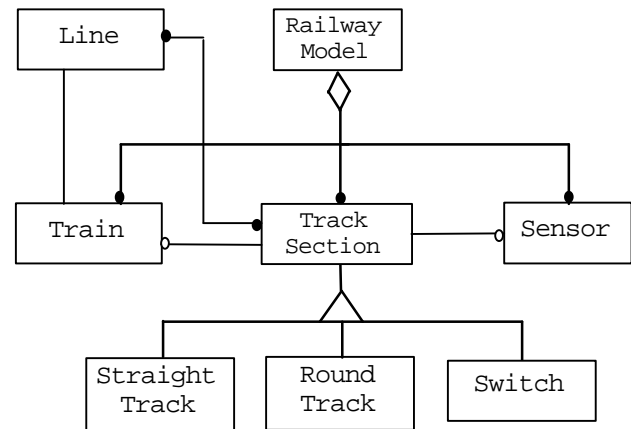


Figure 3: OMT Diagram of Railway Model

## Presentation

Each team has to make a short presentation of the project to all participants of the practical training. Its aim is to improve the presentation skills and to serve as a learning forum for other teams. It normally includes the explanation of requirements, subsystem analysis and designs, nonstandard techniques used, lessons learned and demonstration of the prototype. It does not represent the design review.

## Acceptance Test

After completing the project the team delivers the project results to the supervisors and customers. Individuals from other project teams serve as customers. The acceptance test is quite formal. The project team prepares an acceptance report consisting of a list of final project deliveries and a list of requirements to be validated. The report has to be signed by all parties involved. The project may be accepted conditionally, i.e. some rework has to be done.

Quite often the acceptance test reveals problems to the teams as they don't realize the importance of this

project phase and do not prepare it well enough. Seeing it again in other projects improves it considerably.

## Participants Assessment

The projects are assessed continuously. 50% is the minimum mark to pass the training successfully. If the participant reaches considerably less than the minimum, he/she has to repeat the whole training in next semester. As most participants are well motivated there is normally no need for repetition.

Individual as well as team work is assessed. The quality of the team work carries 50% of the total mark. Another 50% is awarded for individual preparation, constructiveness, and coordination with other team members.

Practical training results are added to the mark in the written software engineering examination. The participants are advised not to try to achieve the maximum in practical training, because it takes an excessive amount of work to score more than 90%.

## Evaluation

The preparation of SEPT is a continuous and time-consuming activity. This is due to changes in software engineering approach during the last few years. Four years ago all projects were developed using structured procedural methods. Now the transition to object-orientation is almost complete. Projects without timing requirements were converted first. The transition started with coding and design. During that stage structured analysis with object-oriented design and programming were used. At the moment conventional testing methods are still used. The efficiency and optimization techniques at design and at code level are becoming more important.

Participants' career opportunities are greatly enhanced by the development environment which includes mechanical, electrical and computer hardware, software and documentation. The influence of industry and technology transfer into industry seem to be very important for further successful training development.

## Future Developments

Already much attention is paid to the reusability of designs. Idioms [8] and object-oriented design patterns [9] are becoming an important part of the training. They increase the productivity and reliability of software enormously. Several well known design idioms and patterns, e.g. Handle/Body, Composite, etc. are in the

stage of introduction. Recently invented real-time design patterns [10] are becoming part of real-time projects.

The use of purchasable class libraries is limited to the standard I/O library. The container class libraries are needed but were avoided in the past because of lack of standardization. With the emergence of STL - Standard Template Library and its standardization the use of an STL implementation is planned.

Problems arose with changes in the provided documentation. The maintenance of paper files proved to be very error prone and time consuming. The provision of some documents as HTML files on the laboratory Intranet since early this year seems to solve many problems. More experience with design and provision of paperless documentation is still needed.

## Acknowledgements

The staff and students of the Laboratory for Microcomputing and Software Construction made Software Engineering Practical Training possible. Peter Stumfol's assistance during the last years was invaluable. Andreas Singer and Eckhart Beck constructed reliable mechanics, electronics and basic software for the railway model. Marcus Bachmann and Paul Heger provided a robust real-time operating system MEBS. Thomas Merkt, Christian Cartus and Bernadin Katic developed software for intuitive visualisation and computer interconnection.

Financial support is from the Ministry for Science and Research of Baden-Wuerttemberg. Hans Jetter and Martin Rieger have provided support and encouragement for many years. Beate Commentz-Walter organizes inspection sessions for developed software. Sati McKenzie from the University of Greenwich made constructive comments on this paper.

## Link:

- [Editing and simulation of a railway model](#)  
FORMAT: TAR+GZIP  
PLATFORM: HP 9000/7XX with HP-UX 9.05  
INSTALLATION:  
gzip -d TRAINSIM.TGZ; tar xvf TRAINSIM.tar

## References

1. Informationen ueber den Fachbereich Technische Informatik der Fachhochschule Albstadt-Sigmaringen, 1996; email: tin@fh-albsig.de
2. Lauber, R.: Prozessautomatisierung I, Springer 1988
3. Jovalekic, S.; Knittel, P.: Objektorientierte Softwarekonstruktion fuer Echtzeitsysteme, Congress Proceedings Echtzeit'95, ISBN 3-924651-46-9, p. 13-21.
4. Jovalekic, S.; Stumfol, P.: Anleitung zum Entwurf, Projektmanagement und Qualitaetsicherung von Projekten, FH Albstadt-Sigmaringen 1994
5. Awad, M.; Ziegler, J.; Kuusela, J.: Object-Oriented Technology for Real-Time Systems: A Practical Approach using OMT und Fusion, Prentice Hall 1996
6. Taegyun, K.: OODesigner V.1.4.1, Pusan University of Foreign Studies, Pusan, Korea 1995
7. Klein, M.H. et al: A Practitioner's Handbook for Real-Time Analysis, Kluwer Academic Publishers 1993
8. Coplien, J.O.: Advanced C++ Programming Styles and Idioms, Addison Wesley 1994
9. Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J.: Design Patterns, Elements of Reusable Object-Oriented Software, Addison Wesley 1995
10. Dietrich, D; Jovalekic, S.: Konstruktion wiederverwendbarer objektorientierter Echtzeitsoftware unter Verwendung von Entwurfsmustern, Kongress Echtzeit'96, Karlsruhe 18-20 Juni 1996