Reverse Engineering: the Challenge of Large-Scale Real-World Educational Projects

Klaus Bothe
Department of Computer Science
Humboldt University Berlin
bothe@informatik.hu-berlin.de

Abstract

The contributions of this paper are twofold: First, a reverse engineering (RE) project is described while other educational projects based on existing systems have mainly focused on maintenance. Second, by means of an RE project, it became possible to approach real-world conditions more closely than in other types of educational software engineering (SE) projects.

1 Problem description

During the past decade, university teachers have recognized the importance of team-oriented projects based on *existing software* [1, 3, 4, 9, 10, 11, 12], which has to be studied, evaluated, modified and extended by students. Above all, arguments are provided to show that maintenance tasks are more typical than the development of software from scratch, in particular for practical work of software engineers in industry.

To prepare students for their professional challenges, several institutions offer appropriate software maintenance classes as a project: Projects at Carnegie Mellon University [4] have been based on software previously developed by students at predecessor courses. At Lund University (Sweden) [12], a software project has been established based on an existing system for telecommunication services, which had to be extended by new services defined by the department's staff. The goal was to build a running prototype on a real telephone exchange with special emphasis on planning activities. At the University of Twente [10], a real-world hospital information system had to be analyzed by the students with the intend to modify requirements and to devise a subsystem structure. The Pascal-S compiler of about 4000 LOC [9] was subject to modifications to support new language features at the Universities of Arizona and Minnesota. Finally, at the University of Western Ontario, they offered several maintenance projects originating either from requests by clients of two companies or from proposals by the university staff to add features and improve the performance of GNU programs [1].

Considering these and similar projects, several *limitations* compared to real-world projects have become obvious:

- Though the original requirements of existing software may stem from real clients, the requirements for maintenance tasks of a project often originated from requests of an academic rather from real clients. Hence, the students considered the project task to be artificial rather than real-world, which may have resulted in motivation problems. For example, Pierce [9] mentioned the effect of a "code-and-fix-then-throw-away mentality", which would be unthinkable when confronted with requests of real clients.

- Normally, for educational projects only a prototype had to be developed rather than a production version. Or, if a production version was requested, only an incomplete version was delivered in the limited time of a class [1, 4]. Bruegge [4] established that "teachers must learn how to deal with incomplete but not incorrect homework."

In addition, none of these projects are reverse engineering (RE) projects in its essence when considering the definition of reverse engineering as "the process of analyzing a subject system ... to identify the system's components and to create representations of the system in another form or at a higher level of abstraction ..." [2]. Instead, maintenance activities in their different forms (modification, extension, porting, bug removal) have been the students' tasks.

In the Fall of 1998, we initiated an educational software engineering project at the Department of Computer Science of Humboldt University Berlin, which still exists and evolves. It goes beyond the educational projects based on existing software [1, 3, 4, 9, 10, 11, 12] in two respects:

- The project has been a proper RE project rather than only a maintenance project.
- It is a real-world project in several respects: There is a real client, the software is in demand, the project has a high priority, the efforts aim at a production system, the problem and the software have a high complexity.

Both aspects belong together: An educational project with real-world characteristics, as in our case, would never have been successful in a project in which the software had to be developed from scratch. Only an RE project could give the students the impression of real-world conditions as well as the satisfaction to supply a quality software product. Similar observations with respect to from-scratch projects are reported by Pierce [9] (resulting in disasters) or by Bruegge [4] (incomplete results).

In particular, the following characteristics concerning our project have to be emphasized:

- The basic situation was a typical RE one: Besides the code, there were no additional documents, and the developers had not been available anymore.
- In addition to RE, the project goals also covered reengineering tasks (e.g., restructuring, code improvements), as well as, maintenance tasks (user requests: modifications, extensions, bug removal).
- The application domain is outside system programming: control software for physical devices.
- The legacy system, as well as, the project requirements stem from requests of real client's (physicists).
- The ultimate goal was a production version rather than just a prototype.

The whole project was a challenge in terms of its size, real-world requirements, a production version, the unknown application domain, lack of developers, and lack of documents. The decisive success factors to overcome this challenge have been the following ones: a proper RE strategy, a dedicated project organization, project reuse and evolving project maturity, and, last but not least, the motivating power of real-world requirements from actual clients.

The purpose of section 2 is to provide a sound basis to characterize the properties of real-world educational projects. The remaining sections discuss the project and our findings in more detail.

2 Characteristics of educational project courses

Educational project courses on RE are more suitable to establish a real-world project environment than forward engineering (FE) projects. Before discussing this thesis in depth (see section 9), let us focus on the basic question: What are the characteristic properties of a real-world educational project?

A number of reports on real-world educational projects exist [4, 6, 8, 10, 12]. Our focus centered around the following questions: To which extend can educational projects with all their restrictions resemble real-world conditions of project work? Which properties are significant in this resemblance?

For an attempt to answer these questions, Figure 1 summerizes characteristics of educational SE projects discussed in a number of different papers [1, 4, 6, 7, 8, 10]. In the left column, the characteristic factors are collected. In the right column, possible properties (values) of these factors are fixed.

A project approaches real-world characteristics by possessing as many attributes at the right side as possible. Specifically, requirements should stem from real-demands, the application domain should be outside system programming (unlike compiler construction and operating systems), there is a real client (not just a lecturer), the goal of the project should be a production version, the project has a high priority, there are project deadlines, it is a large-scale project, and all phases and the respective documents are covered. Figure 1, above the horizontal line, applies to general SE projects. The last three factors below the line apply to projects based on existing software. Thus, we distinguish the source of original requirements of the existing software from the source of project requirements.

(Remark: Figure 1 includes the factors which are inherent to projects. Factors of project management like planning or tool usage are not considered.)

Our position and our experience is the following one: The closer we approach real-world attributes, the more the students are motivated, the higher the learning effect, and the better the overall success of the project.

factors	attributes				
requirements' source	academic		-		real demand
application domain	system programs		-		other
client (customer)	lecturer	-	non-comm.	-	commercial
goal project status	prototype		-		production version
project priority	low		-		high
deadlines	no		-		yes
project size	low	-	medium	-	large scale
phases (and documents)	only selected phases		-		all phases
original requirements' source	academic		-		real demand
project requirements' source	academic		-		real demand
task class	maintenance		-		reverse engineering

Figure 1. Characteristics of educational SE projects

For educational RE projects, nearly all real-world attributes can be covered. The most serious problem for an educational site stems from deadlines which cannot be guaranteed by student projects.

Figure 2 applies the general characteristics of educational SE projects to a couple of particular projects based on existing software. The last column refers to our project at Humboldt University Berlin. Positive attributes relating to real-world projects have been underlined.

factors	Wohlin 97	Pierce 97	Sikkel 99	Andrews 00	HU 00
original requirements	<u>real dem.</u>	<u>real dem.</u>	academic	<u>real dem.</u>	<u>real dem.</u>
project requirements	academic	academic	academic	<u>real dem.</u>	<u>real dem.</u>
application domain	<u>other</u>	system prog.	<u>other</u>	<u>other</u>	<u>other</u>
client (customer)	lecturer	lecturer	lecturer	<u>commercial</u>	<u>non-com.</u>
goal project status	prototype	prototype	prototype	prototype	<u>prod. vers.</u>
project priority	low	low	low	low	<u>high</u>
deadlines	no	no	no	no	no
project size (in LOC)	<u>large</u>	<u>4000</u>	<u>large</u>	<u>large</u>	<u>27.000</u>
phases	<u>all</u>	<u>all</u>	requ., des.	<u>all</u>	<u>all</u>
task class	maint.	maint.	maint.	maint.	<u>RE</u>, maint.

Figure 2. Characteristics of selected projects based on existing software

3 A brief introduction to our project

How the project started

During the summer of 1998 we received an email from a professor of the Department of Physics of Humboldt-University Berlin with the following request:

> *" ... I turn to you for advice. We are a working group in the field of X-ray defraction on layer systems. In our laboratory, 10 PCs are controlling X-ray topographic cameras. Some years ago, a software engineer had developed a C++ program for Windows 3.11. Unfortunately, the program is not very readable. My question: Could you give us advice how to gain a sound software base and how to maintain it with a reasonable effort?"*

To better understand the initial state of the legacy software and of the client's situation we answered with the following catalogue of questions, which may also prove useful in similar RE projects (Figure 3).

This was the very beginning, and after first direct contacts with the physicists we started a project course during the Winter term of 1998/99.

Application domain

Our subject system had to prepare, to control and to evaluate physical experiments based on a so-called X-ray topographic camera. Several drives move a sample which has to be investigated by means of X-rays. The software has to move the drives, evaluate the values measured by detectors, control long-term measuring processes, provide for special-purpose graphical representations of the results, etc.

1. Is the software engineer still available?
2. How well is the program documented or commented?
3. Are there any further documents: user manual, specification of requirements ... ?
4. What is the size of the program in LOC?
5. Which C++-system is the basis of your development?
6. Is there any special design method you applied (e.g. OO design, structured design)?
7. What are your problems: errors, stability, porting, extensions ...?
8. What do you think is the right way to proceed: improve the system or rewrite it?
9. When do you need the results?
10. Who should maintain the software in the future?

Figure 3. Catalogue of questions at the beginning of our RE project

Project goals

To be successful, our project could not only rely on pure RE methods to identify subsystems and to create representations of the system at a higher level of abstraction. A pure RE project would not have succeeded since RE activities alone would never have satisfied the client. RE activities alone are of little or no interest for a client. However, the client is required as the domain expert. Thus, we identified three lines of project goals:

- *Reverse engineering* of the system:
  - development of missing documents (specification of requirements, user manual, design, test cases ...)
  - architecture recovery
  - program understanding (commenting the sources)
- *Maintenance* (fulfillment of *client's requests*):
  - stability of the system
  - extensions: add new devices to the system
  - porting to an actual operating system
- *Educational goals:* team work, project management, real-world project, production version, application outside of system programming, contacts with real clients ...

Challenge and approach

The challenge of our subject system was its size, which was large-scale for educational projects. Furthermore, the developers and documents were missing, the application domain was unknown, and the target system had to be a production system rather than just a prototype. Facing these problems, there was the obvious question: Is it reasonable at all for an educational site to take the responsibility for such a system and its enhancement or is it a too ambitious task?

However, this challenge also had a positive aspect: It involves a great chance for educational graduate projects corresponding to our position: The closer we approach real-world attributes, the better the students' motivation, the learning effect, and the overall success of the project.

Faced with the challenges described above, the following factors were decisive for the project's success: the RE strategy, a dedicated project organization, project reuse and evolving project maturity, a problem-oriented decomposition of the task, and, last but not least, the motivating power of real-world projects. Figure 4 summarizes the factors influencing the student's motivation.

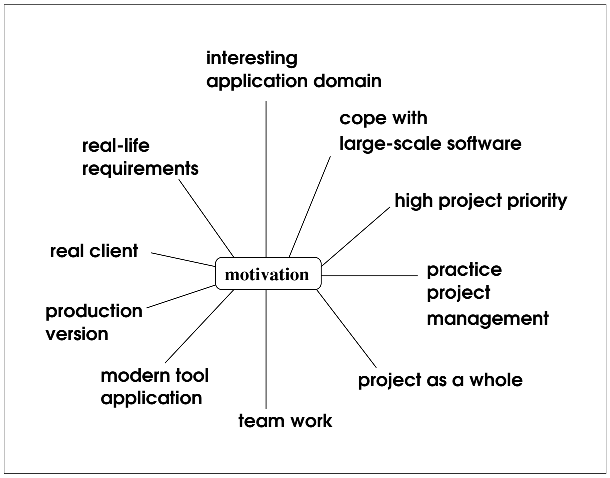

Figure 4. Motivating factors

Curriculum's background

At Humboldt University Berlin, full-time students should obtain a Master's degree after 9 semesters (4.5 years). In reality, it takes longer. After the first two years, the first phase is completed, which is composed mostly of compulsory courses in the major subject area (Computer Science). During the second phase, students have a considerable degree of freedom of choice of advanced courses. However, there is no formal division within the 9 semesters of Master's degree between an undergraduate (Bachelor) and a graduate (Master) curriculum - the Bachelor does not exist as a separate degree.

In the second phase, the students are offered a class on "Introduction to SE", accompanied by simple exercises. Afterwards, they are free to attend basic or even advanced projects to apply what they have learned in the SE lecture. At the end of the degree program, the students write their Master's theses.

It was our idea to integrate the basic project, the advanced project, and the Master's thesis into a sequence of (optional) steps into our project work.

4 Applied reverse engineering strategy

In the literature on RE, a lot of techniques and tools have been discussed depending on the original state of the legacy software, as well as, on the tasks to be fulfilled [2]. What can be done in an educational project?

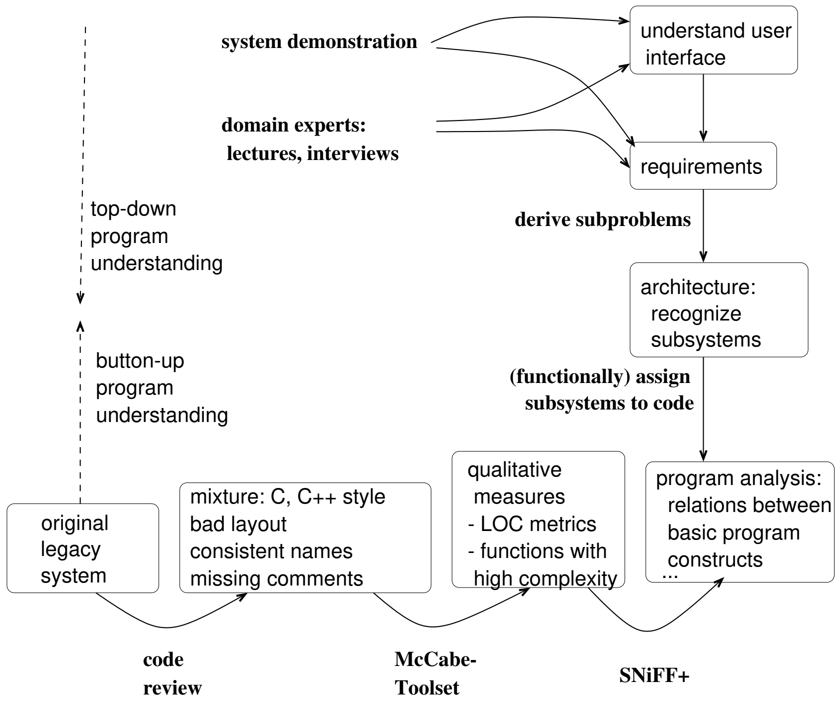

Figure 5. Steps of our RE strategy

Figure 5 summerizes the steps of our RE strategy (bold faced words at arrows) and the results of these steps (in boxes). RE is restricted to analysis of the software. Our analysis has been driven from both ends: In a top-down approach, we tried to understand the program starting with the external behavior of the system as demonstrated by the physicians in their laboratories. In addition, the domain experts offered lectures and gave interviews on the application domain. On the other side, the original legacy system was investigated in a bottom-up approach by code reviews and by tools (McCabe tool set for metrics on LOC, cyclomatic complexity, and object orientation; SNiFF+ for the analysis of basic program structures). Several tools could not be applied for different reasons (e.g. some tools only support C but not C++).

5 Multi-semester projects with different levels of expertise

It is well-known that large-scale educational projects cannot be finished within one or two semesters [7, 9]. Consequently, we derived our basic project organization as a (potentially) open-ended project with tree levels of expertise (Figure 6): basic, advanced, and master.

Every second semester, an introductory project was offered followed by a permanent advanced project course, which allowed us to recruit subjects for master theses. In this final expert level, usable results for the client were produced from time to time. Hence, the client received valuable feedback for his efforts.

It was not necessary to limit the number of participants a priori. Up to now, both introductory courses started with 9 students. Currently, we have 15 students in the advanced project in the forth semester.

The team structure had been rather heterogenous since the team consisted of participants

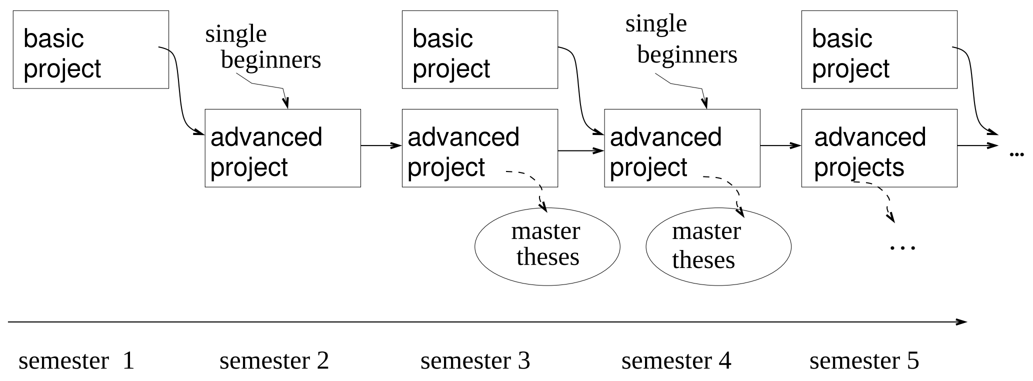

Figure 6. Multi-semester projects with three levels of expertise

at different phases within the curriculum (ranging from the 5th to the 10th semester at the graduate level). Consequently, their knowledge as well as the actual educational goals differed considerably.

6 Project reuse and evolving project maturity

Project reuse has been discussed by different authors [4, 9, 12]: The supervision of academic project courses is expensive for the university staff.

Bruegge [4] points out that projects require a large investment by the instructor to set up the infrastructure for the course. Hence, this infrastructure should be reused. The Pascal-S compiler maintenance project course [9] had been reused at least from 1988 to 1997.

In particular, large-scale projects dramatically increase the effort for the instructors. In our project, this effort covered the following areas:

- organizing the contact with the client
- learning about the application domain (to assess the requirements documents)
- assessing the legacy program (is it appropriate for a project course?)
- project management (monitor the students' activities)

We applied a special kind of project reuse: Our project had been reused from semester to semester with an increasing level of maturity. It evolved rapidly over the semesters, and each subsequent class built upon the work of the previous class. For example, in the field of requirements specification, documents for new subtasks of the system were added, and existing documents were replaced by improved versions.

7 Decomposition of tasks: role-oriented vs. problem-oriented

One basic problem in the organization of group projects is posed by the assignment of individual tasks to a student. Most educational projects employ a role-oriented assignment of tasks to project participants. For example, Pierce [9] and Wohlin [12] introduce the roles of a team leader, a configuration manager, a quality assurance manager, a maintenance engineer, and a verification engineer.

To support open-end projects and to be flexible in task decomposition, a role-based approach is not appropriate for a large-scale RE project. It assumes that different project partners work in parallel during the same period of the project (semester). A problem-oriented decomposition is much more suitable for multi-semester course work: Each semester, one (or several) of the subproblems can be addressed by one student (or a group) for all phases. In addition, problem-oriented task decomposition is more interesting and educational for students because of their involvement in several phases.

Thus, we basically assigned the students to subproblems according to a use-case-oriented approach (Figure 7). For such a subproblem (corresponding to a use case), the requirements document, the design, the code and the task of restructuring the code had to be considered by the corresponding students. However, for some special roles, we applied a role-oriented assignment: testing, applying metrics, porting, and the handling of version control.

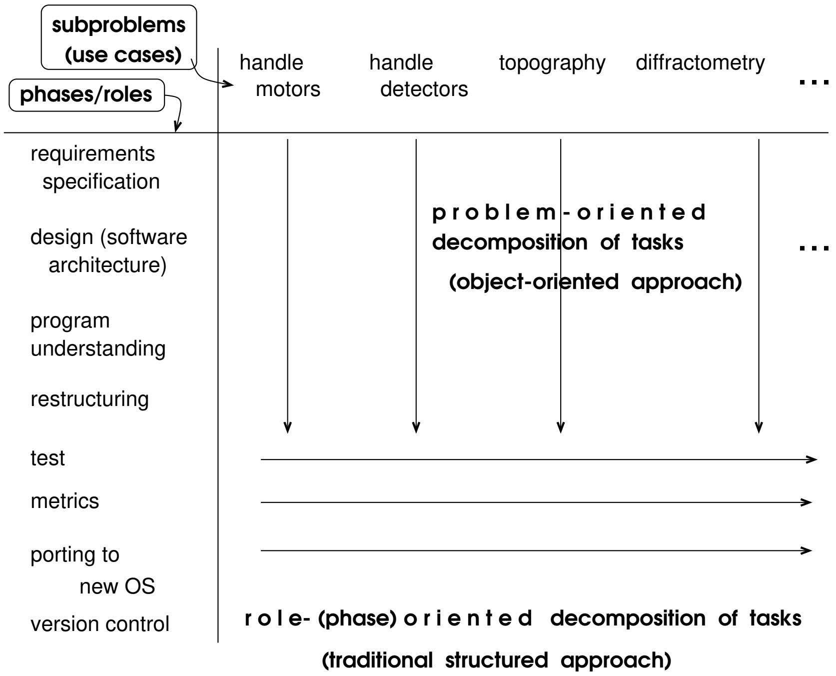

Figure 7. Decomposition of tasks

8 Ordinary project management applies to RE projects

Not surprisingly, large-scale RE projects require the same approved techniques of project work as ordinary projects from scratch do. In particular, the following details of our project management shall be emphasized:

- Each week we organized a project *meeting* to discuss common problems, as well as, present particular results by special groups.
- *Protocols* (by meetings and individual discussions) are very important in large-scale group projects in order to remember commitments and to inform everybody about them. During each meeting, another student had to write the corresponding protocol.
- For all documents developed in an RE project, we need *guidelines* to provide the students with reference points for their orientation and to ensure a unified document style.

- Our *project repository* covered requirements documents, design documents, as well as, code and test documents in different versions. It was decisive for a multi-semester project not to lose information and to introduce new students to the existing knowledge base in basic courses.

9 Superiority of RE over FE students' projects

In section 2, we considered characteristics of educational project courses. Our thesis was that RE projects are more suitable to meet real-world project characteristics in an educational environment than FE projects. Now, after collecting experience in an actual RE project, more details can be given to stress the validity of this thesis:

- In RE projects, students are immediately confronted with large-scale software. In contrast, FE projects take a longer time before a large piece of software has been developed.
- Building large-scale software *from scratch* takes considerable effort and time: In an educational environment, these projects generally end with disastrous results or remain incomplete [4, 9].
- During the initiation of an RE project, we already have a working system covering all the intended system functions.
- Legacy systems represent considerable man power performed by other persons (system engineers), including, for example,
  - implementation strategies and techniques, as well as,
  - testing effort guaranteeing a certain amount of stability.

  This effort can be reused by RE projects, thereby, reducing the coding and testing pressure.
- Legacy software represents a framework for a software solution to an application domain. Studying this software, students can learn about this solution in an efficient manner.
- In summary, our RE project covered nearly all the real-world attributes posed in Figure 1 with only one exception: We cannot guarantee deadlines, which is in the nature of educational projects.

10 Problems with large-scale real-world educational RE projects

This section summarizes problems we have faced with in our project. Some of these problems are typical of RE projects while others also apply to general real-world educational projects.

- Having had a first code review of the system, the students suggested that the system might simply be reimplemented to avoid all the bad properties of the legacy system. We argued with the general advantages of RE projects in terms of saved time and man power [2].
- RE is an activity of little interest to a client such that one cannot restrict such a study to pure academic goals limited to RE. This study has to be combined with maintenance tasks corresponding to user requests.

- Hence, in our first project phase, our client did not believe that we were performing any work at all. Before being able to maintain the software, we had to invest into an extensive RE phase to understand the large-scale program.
- The main problems the students faced concerned the complexity of code (program understanding) rather than the complexity of the whole project. The requirements and design documents had been developed by the students without much difficulty. However, assigning functional requirements to dedicated portions of legacy code or understanding particular code fragments took much more effort.
- To develop a production version in a technical system turned out to be very hard in an educational environment. The problems of correctness and stability had been underestimated by the staff: Only one student was responsible for testing in the first project phase.
- To treat the domain experts with care, we need a staff member acting as an intermediate between the students and the domain experts. Otherwise, domain experts may easily be overwhelmed by excessive inquiries of students, who, for example, repeat the same question over and over.
- In a large-scale legacy system, different users have only a restricted view of the system's behavior such that several user groups had to be included.

11 Summary

Finding projects appropriate for educational maintenance and RE projects poses a challenge [1]. Due to the nature of the request of our academic partner, we believe to have found a rather ideal real-world project for educational purposes. The following key ideas of our article shall be summarized:

- In an educational environment, large-scale RE projects are more likely to succeed than projects from scratch.
- The character of RE projects is rather different from maintenance projects: RE projects focus on all documents of the legacy software system (requirements, design, code, test) while maintenance projects focus on modification, extension, porting and bug removal.
- In our experience, the main factors of success are:
  - multi-semester project
  - three levels of project participation: introductory, advanced, master
  - extensive reverse engineering phase (without modifications of the code, in order to understand the code, the structure, and the requirements of the software)
  - motivation: real-world requirements, real-world client, interesting application domain, learning about project management, and others.

References

[1] J. Andrews, H. Lutfiyya: Experience Report: A Software Maintenace Project Course, 13th Conference on Software Engineering Education and Training, Austin, Texas 2000, pp. 132-139.

[2] R. Arnold: Software Reengineering, IEEE Computer Science Press 1993.

[3] S. Brown, N. Wilde, J. Carlin: A Software Maintenance Process Architecture, 9th Conference on Software Engineering Education and Training, Daytona Beach, 1996, pp. 130-141.

[4] B. Bruegge: From Toy Systems to Real Software Development: Improvements in Software Engineering Education, SEUH'94, pp. 62-72.

[5] A. Dingle, T. Hildebrandt: The Maintainability Gap, 13th Conference on Software Engineering Education and Training, Austin, Texas, 2000, pp. 208-209.

[6] J. Harrison: Enhancing Software Development Project Courses via Industry Participation, 10th Conference on Software Engineering Education and Training, Virginia Beach, 1997, pp. 192-203.

[7] H. Mayr: Teaching Software Engineering by Means of a "Virtual Enterprise", 10th Conference on Software Engineering Education and Training, Virginia Beach, 1997, pp. 176-184.

[8] W. McMillan, S. R. Aprabhakaran: What Leading Practitioners Say Should Be Emphasized in Students' Software Engineering Projects, 12th Conference on Software Engineering Education and Training, New Orleans 1999.

[9] Pierce: Teaching Software Engineering Principles using Maintenance-Based Projects, 10th Conference on Software Engineering Education and Training, Virginia Beach, 1997, pp. 53-60.

[10] K. Sikkel, T. Spil, R. van de Weg: Replacing a Hospital Information System: an Example of a Real-World Case Study, 12th Conference on Software Engineering Education and Training, New Orleans 1999.

[11] J. Slimnick: An Undergraduate Course in Software Maintenance and Enhancement, 10th Conference on Software Engineering Education and Training, Virginia Beach, 1997, pp. 61-73.

[12] C. Wohlin: Meeting the Challenge of Large-Scale Software Development in an Educational Environment, 10th Conference on Software Engineering Education and Training, Virginia Beach, 1997, pp. 40-52.