Teaching Software Engineering Through Real-Life Projects to Bridge School and Industry

Ki-Sang Song
Dept. of Computer Education
Korea National University of Education
kssong@knuecc-sun.knue.ac.kr

Abstract

To educate graduates to succeed in industries which demand high quality software engineers is not easy due to rapidly changing organization styles and working environments. The major limitation of university education may be the lack of opportunity to expose students to real field problems. In this article, we present our experience of exposing graduate students to a real-time plant monitoring and control software development project and show how the software engineering process has been customized to educate them and satisfy the user requirements at the same time.

1. Introduction

Unlike the environment of research institutes and computer science related departments, graduates may be confronted with product-oriented jobs at their work. Therefore, it is not easy to educate graduates to succeed in software industries which demand high quality software engineers due to rapidly changing organization styles and working environments. The major limitation of university education may be the lack of the opportunity to expose students to real field problems.

When educating students with real projects, several issues such as management issues, technical issues, and project phases control issues need to be handled simultaneously. In school education, I believe that the management issue is more significant than any other issue due to the difference of culture in school and industry.

Therefore, it is required to customize the software engineering education in accordance with the student experience level. This does not mean that some software engineering processes should be deleted. Instead of eliminating some process, certain parts of them must be stressed for educational purposes.

In this article, we present our experience of exposing graduate students to a real-time plant monitoring and control software development project and show how we have approached to pursue two objects: to educate students and to satisfy the customer's requirements of the final product.

2. Project description

The project, sponsored by LG(Lucky Goldstar of Korea), was to develop a real-time factory monitoring and control system, called "EYE-2000". It can monitor and control thousands of analog and digital points, has a reliable multichannel communication protocol between processors, and can display process variables in multiple, concurrently updating windows on the same CRT screen. The detailed specifications of EYE-2000 are described in Table 1.

EYE-2000 uses an off-the-shelf microcomputer, IBM 586, and Microsoft

Windows 3.1 and can possibly be run on a multitasking operating system such as Windows 95 or Windows NT.

Table 1. The specifications of EYE-2000

Number of analog signal points	500
Number of digital signal points	1,000
Number of loop control points	50
Connected peripheral devices	16 PLCs, SLC, Laser printer, Dot printer
Number of windows on a screen	50
Number of reports	30
Number of displayed event data in a window	20,000
Number of displayed alarm data in a window	200
Sampling time of process variables	every 2 seconds
Process variable values to be saved for analysis	6 sec, 1 min, 6 min, 1 hr, 24 hrs average value

EYE-2000 is composed of three main parts: communication, process variable monitoring, and report generation for operators. Data from multiple points can be displayed on the PC text screen in real-time, using multiple windows. PC and proprietary PLCs communicate to update new data every 2 seconds. Also, reliable communication is crucial in factory control and we need to use very reliable protocol.

Graphic user interface(GUI) is essential in processing variable monitoring software for user-friendliness. The user can open up as many as 16 windows, make them any size, and position them anywhere on the screen. The graphics inside each window are completely controlled by the operator.

The report generation function is required for keeping plant operating records. When events are detected by a sensing/input device of a control system, the event is recorded in the event database history files and can be reported to operators according to the schedule.

When LG suggested this project to us, they agreed to prepare the specifications and provide relevant information if requested. Also, they agreed to survey customers, particularly factory operators, potential users of EYE-2000, to give information for window design. However, provided information was superficial and only useful for guidelines of the final product. Therefore, we had to organize a project team before we could commence the software engineering process.

Even though the clients understood that school is not a profit-oriented software engineering enterprise, they expected that this project to produce a good seller. Also, time spent for developing software was critical to them. Thus, we had to take care to perform every steps in the software engineering.

3. Issues of engaging in software engineering with students for real-life projects

EYE-2000 is more than 100,000 lines of coded software. To expose students to such a large project and to provide them hands-on experience, several issues should be considered. Software engineering education may heavily focus on technical aspects. But they already have attained sufficient knowledge from well-handling organized classes. They only lack handling the real project utilizing their school education. Under these assumptions, I tried to educate students by providing them industry-like working environments.

1) Organizing project team

Team dynamics can significantly affect the completion of the whole project process. For that reason, team organization is very important for the success of software engineering. There are various ways to organize software development project teams with students, such as random selections and the teacher chosen approach[1].

At the project planning stage, I thought only three graduate students would be enough to complete this project with eight months development time. Based on this estimation, I planned to organize the project team with two graduate students and one external person. Two students had a solid background in data structures, database systems, operating systems, algorithms, Borland C++, and object-oriented programming. Also, they knew Microsoft Windows programming, OWL, resource workshop, and DDE, in addition to software engineering.

The rationale of including an external person was to practice the interpersonal communication skill which is required for team work. By working with people who have different backgrounds, they may practice communication skills for cooperation. The selected external person had some experience in Windows programming. He was not a professional software engineer, but only had a little computer science background.

This kind of team organization might provide very similar industry culture for students and they acknowledged this working environment was helpful. Even though they did not open their minds well at first for exchanging information to solve problems or know-how of utilizing developing tools, about one month later they cooperated very well. Each student was implicitly assigned as the project manager for partitioned problems. Though I was really responsible for the whole project, I planned to show students how to manage a real-field project.

2)Management issues

The traditional software development life-cycle considers every phase of software engineering, such as problem analysis, system requirement establishment, design of solution, coding and debugging, implementation, documentation, testing, and maintenance. Instead of applying the conventional approach, we used the object oriented technique to model and implement the EYE-2000.

System analysis is the first step of the object modelling technique methodology[2] and students did practice to abstract object models, sequencing of interaction and data transformation relations between objects. Since the system development environment was Microsoft Windows, many objects such as "window" and "dialog box" had been considered for object modelling.

(1) How to communicate with clients?

In most cases, students are generally familiar with school lab projects which are well-organized and well-defined. Therefore, the difficulties of figuring out the clients requests in object modeling appealed to the students. Since system analysis is not a mechanical process, and most statements of problems provided by clients may lack essential information, it is crucial to extract such information either from the requester or the analyst's knowledge.

The project team had met several times with industry engineers and tried to elicit what users wanted. I tried not to intrude into their discussions unless they were stuck either for a technical problem or adjusting system specifications. In the beginning, students attended and only observed the meeting processes to exchange information for making specifications of the final product. They sometimes gave their ideas of implementation problems at that stage. I wanted to show how students can figure out system design specifications from discussions with clients. After each meeting, they were requested to present their opinions and do system analysis based on the meetings. As meetings continued, I did not attend and students actively discussed

with industry engineers for system analysis. From these meetings and system analysis steps, students could acquire the knowledge of the user's present job[3] and eventually gain the knowledge and understanding of the target users.

(2) How to encourage students?

Since EYE-2000 employed many graphic user interface screens, the clients wanted to survey real users for convenience of screens to users as soon as possible. Therefore, we decided to apply the rapid application prototyping technique[4] instead of following the step-by-step conventional software development life-cycle.

Prototyping is usually used for producing very quickly a working version of a piece of software[5]. The main purpose of prototyping is to reduce the long time gap between the user acceptance phase and system analysis phase in conventional life-cycle. The EYE-2000 development deadline was very crucial to clients as aforementioned. Therefore, if there existed a long time gap, we thought it would be difficult to do massive reconstruction of the final software. With these two constraints, the project team had to learn how to prototype the EYE-2000.

The main difference between a prototype and the final desired system was its performance, thus we omitted the PC-PLC communication part and user report generation part. A prototyping monitoring system only was tried by system specifications, and we used simulated data for observing process variable trends. Normally, a prototype system can be created using whatever tools the developers may be familiar with. We used operational techniques derived from object modelling and functional models between objects.

As the main purpose of prototyping a system is to aid the analyst and design stages of a project, we could discuss this with the clients and project team. By enabling users to see very early what the system could do, they could give more helpful advice for window designs. This was especially important for our project due to the user-friendly request emphasized for the EYE-2000. If the EYE-2000 designed and implemented only the developers point of view, then the users might complain about some window design based on the user-friendly aspect. Therefore, we applied the prototyping technique for the EYE-2000.

Another reason for applying the prototyping method stemmed from psychological purposes. It is quite general that every engineer may feel fear when they meet a new project. In addition to this aspect, students felt more stress because they did not have experience with the plant monitoring and control process.

Although we spent much time studying real-time factory operations, the project team looked as if they were not sure how to start. Since this project was the first time for them to be involved in a real-life project, they showed hesitation to dare to solve the problem at first. Even though they had a sound background and knowledge of software engineering and computer science as aforementioned, they still felt fear of what they must solve.

I felt that just documenting the specifications was not enough for these students due to their lack of the real-life problem experience. Thus, I applied the rapid prototyping method to overcome hesitation and fear of failure for the real-life problem. Through prototyping they could see what would be done through this project. Also, this assured that they might vividly see what they could do through prototyping the EYE-2000. By prototyping the system, even though in abstract

style, it was enough to show them what the final outcomes would be. The students were finally able to see the solutions. I believed this is essential for them to have self-confidence before getting into an industrial position.

This prototyping job was approached with systems specification. Since students already acquired enough knowledge of system by discussion with industry engineers, they were able to finish the prototyping within one month.

(3) Training to get knowledge from an unfamiliar field

For system analysis, software engineers usually need much knowledge of the target problem in addition to computer science area information. Some parts can be easily attained, but it usually requires great pains to get them. Most industry jobs request engineers to be familiar with these processes. Through obtaining much knowledge of the target problem, they may design more effective systems.

To practice this process, we asked students to attend workshops and seminars to gain expertise for real-world problems. they were asked to model objects derived from real-world problems, such as PLC communication and timing problems, by applying object modelling techniques. In our experience, attending a workshop may be greatly helpful in acquiring this related information. If they could get the basic information of the target system, they could attain necessary information by study alone.

(4) How to cooperate to integrate parts?

Since EYE-2000 is composed of three main parts, each team member was responsible for communication, process variable monitoring, and report generation subsystems. Also, we developed EYE-2000 using Borland C++ and Object Windows Library(OWL) which is object oriented layers on top of Windows API[6]. Students needed to cooperate in system integration and test processes.

Applying object-oriented techniques, students could save much effort in integrating sub-systems. Object modelling and class designs needed joint effort. It was possible to use the advantages of inheritance for GUI design. But testing over 100,000 lines of coded program significantly required each member's cooperation. We found that sometimes advice from others could solve coding mistakes very easily. With this philosophy, students were encouraged to actively share their knowledge and have an open-minded toward criticism.

5. Conclusion

The main purpose of exposing students to real-life software projects is to provide and experience industry-like environments for easy transition from school knowledge to product-oriented industry jobs. The ways to organize projects team, managing students' fear from large scale real-life projects, improving communication skills with clients, and getting knowledge from an unfamiliar world are practically required points, in addition to software education. By organizing project team with students and an external engineer, we could give students a closely simulated industry environment. To overcome students' fear and hesitation from large and unfamiliar project, we used the rapid prototyping method.

From this experience we should stress that the software engineering process can be appropriately customized to educate students, if it is required, without failing to meet client requests. By customizing software engineering processes according to the needs of students, university software education may provide a more pragmatic real-world working environment for software engineering students and contribute to their success in industries.

References

[1] Thomas J. Scoot et al., "Team Dynamics in Student programming Projects," *ACM SIGCSE*, Vol. 25, No 2, pp. 111-115.

[2] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, *Object-Oriented Modelling and Design*, Prentice-Hall Inc. 1991.

[4] L. Macaulay, "Cooperation in understanding user needs and requirements," Computer Integrated Manufacturing Systems, Vol 8, No. 2, pp. 155-165, 1995.

[4] J. P. Reilly, *Rapid Prototyping Application*, International Thomson Computer Press, 1996.

[5] D. Bell, I. Morrey and J. Pugh, *Software Engineering*, Prentice Hall International

[6] C. Petzold, *Programming Windows 3.1*, Microsoft Press, 1992.

**********Bandwagons From Page 58**********

We need to explore new pedagogical avenues, but we cannot abandon existing topics and approaches without thorough consideration of the intellectual and experiential underpinning they provide. We cannot allow ourselves to become so enamored of our pet approaches that we fail to acknowledge existing alternatives, both old and new. We cannot impose our own version of "pedagogical correctness" on our students. We owe them a balanced presentation in the introductory course, a solid foundation on which their further education can build.

References

[1] Knuth, D. E., *The Art of Computer Programming*, vol. 1, *Fundamental Algorithms* (Addison-Wesley 1968, second edition 1973).

[2] Jensen, K. and N. Wirth, *Pascal User Manual and Report, second edition* (Springer Verlag, 1974).

[3] Abelson, H., G. J. Sussman, with Julie Sussman, *Structure and Interpretation of Computer Programs* (MIT Press/McGraw-Hill 1985).

[4] Dijkstra, E. W., "Go to statement considered harmful," *Communications of the ACM*, vol. 11 no. 3 pg. 147 (March 1968).

[5] Knuth, D. E., "Structured Programming with go to Statements," *Computing Surveys* vol. 6 no. 4 pg. 261 (December 1974).

[6] Pattis, R. E., "The 'Procedures Early' Approach in CS 1: A Heresy,"*SIGCSE Bulletin* vol. 25 no. 1, pg. 122 (March 1993).

[7] Astrachan, O. L. (panel moderator), "Computer Science: The First Year Beyond Language Issues," *SIGCSE Bulletin* vol. 28 no. 1, pg. 389 (March 1996); additional materials available at http://www.cs.duke.edu/~ola/slides/lang96.html

Call for papers
DPLE 1997
First International Conference on
Declarative Programming Languages in Education
September 3-5, 1997
University of Southampton, England

Papers accepted for the conference must contain material not presented previously in any formal forum. Submissions will be judged on relevance, originality, significance, correctness, and clarity. Each paper should explain its contribution in both general and technical terms, identifying what has been accomplished, saying why it is significant, and comparing it with previous work. Authors should make every effort to make the technical content of their papers understandable to a broad audience.It is intended to publish the proceedings in the Springer LNCS series; authors should submit the full paper in Springer conference style. For further details on how to submit a paper please consult our web site at www.ecs.soton.ac.uk/southampton1997/dple/. Deadline for submissions is April 1,1997.