Rethinking PBL as a Holistic Pedagogical Method

Is PBL applicable to develop learners' self-awareness in software engineering education?

Hiroyuki Kameda, Taichi Nakamura, Akio Takashima, Shun Hattori
School of Computer Science
Tokyo University of Technology
Tokyo, Japan
{kameda, nakamura, takashima & hattori}@cs.teu.ac.jp

Abstract— **This paper discusses PBL for first year software education, paying attention to aspects of holistic learning, and also proposes a new PLB course based on the discussion. The PBL course was given to high school students to confirm fundamental validity. The questionnaire method suggested that PBL might be fundamentally useful to learn holistically.**

Keywords-Education; Project-Based Learning; holistic learning; software engineering education; awakening;

I. INTRODUCTION

Recently, software has been a social infrastructure that takes an important role in our daily life. Many kinds of software systems, i.e., air traffic control, network banking, mobile phone, e-commerce, e-learning, ubiquitous home, electronic books, and so on are one after another affirmatively proposed, willingly created, widely distributed, and intensively used all over the world by many people, while many requirement changes are also dynamically required to those systems according to the floating social needs which are subject to social rapid progress. Due to these situations, software industries strongly ask university students to reach practical level, i.e., professional level when they graduate from universities. This request is difficult for university teaching staff to respond to so far.

On the other hand, some countries, e.g., Japan, face other serious problems such as birthrate that has been rapidly decreasing. For this reason, the era is coming soon that universities admit all students' entrance to universities with no entrance examination. This fact is also surely one of the very reasons to accelerate reduction of learning motivation of students and declination in academic abilities in Japan. In these backgrounds in Japan, Japanese software companies strongly demand universities to develop software engineers with a fairly good amount of abilities needed by industry.

But as we know, there is a seriously big gap between what universities teach to students of computer science and what industry demands of them. At the same time, unfortunately the number of young people who want to study software engineering has been rapidly decreasing in Japan, because they do not like mathematics, physics etc. Moreover, the incentive of studying IT in high school students is recently decreasing, especially in Japan. To cope with these serious problems, Nakamura, Kameda et al. [1] proposed and have been studying a new version of software engineering education named "Tangible Software Education" since 2007 with the support of national grant of Japan, where the meaning of the word tangible in this project is defined as if it were visualized enough to be controlled manually through mouse or keyboard. One of products of the project is a new course of software engineering education designed for novice software developers. This course was applied to university students with a PBL (Project-based Learning) method, and fundamental validity was confirmed by educational evaluation (Kirkpatrick's Four-Level Evaluation) [2].

In this paper, we report another application of the new software learning course for novice programmers to high school students who participated in a science summer camp held at our university this summer, in order to evaluate the PBL course from mainly two view-points; one is to know if the course is really valid and the other is to prove its effectiveness as a teaching method of holistic learning teaching for novice.

II. PREVIOUS PROGRAMMING METHODS AND THEIR PROBLEMS

A. Previous programming education for first year students

Usually, first year university students who want to study computer science, especially software, have courses such as courses of computer literacy and programming mainly focused on grammar of programming languages and how to use software developing environments. A few years ago, these courses were useful for students to learn about computers and programming at the same time. But as it was mentioned before in this paper, software is now too large-scale, complex, and changeable to create and manage well. The previous software course is now indeed obsolete.

B. Currently challenging trials and thier problems

To cope with the problems mentioned above, many interesting learning methods have been proposed since around 2000.

Scratch [3] is an example designed for novice programming, e.g., children learning the essence of programming. In Scratch learners write simple programs to control software robots. In this sense, Scratch seems similar to the educational programming language LOGO. But in Scratch, the world

This project is supported by Tangible Software Education and Research as part of the Program promoting the leveling of private university academic research by Japanese government.

where software robots move around is more beautifully visual and more fantastic, so that young learners are attracted. Alice [4] is also the same kind of educational software. In Alice, program is constructed as a sequence of English statements (commands). This feature is effective for every novice programmer to learn what/how programming is intuitively with no complexity. Indeed these educational software are very attractive and seem effective, but only to people who just learn the basics of programming. They are not useful for university students of computer science who should be IT professionals in the future to learn knowledge and skills of software production.

Robocode [5] is an educational software for students of schools of IT. Robocode provides a Java programming environment and learners learn programming by writing simple programs of robot battle games. Robot battle is fun for young boys but some do not like such battles. Moreover, writing programs of Robocode is indeed difficult for most of the novice students. For these reasons Robocode is not appropriate for FYE (First Year Education) in universities.

BlueJ [6] is another type of educational software, which is designed to learn Object-oriented programming in Java, but quite different from Robocode. BlueJ provides a new type of programming learning. Classes are displayed in UML class diagrams, instances of the classes can be displayed in a pane visually, and also Java source codes can be edited in editor windows. For example, functions to show field parameters and their values dynamically are also incorporated in BlueJ to help users figure out sequences of run time actions of programs users are writing. But this educational software is rather difficult for novices to use properly. Easier ones are needed for first year education.

Greenfoot [7, 8] is the very educational software, aiming at realizing an introductory education of high school students. The educational software is designed for programming beginners easily to create programs especially such as games and simulations, based on an educational theory, the Kolb's circle of learning [9]. Moreover it also adopts the object-oriented programming method, so that teachers only prepare software resources, e.g., fundamental classes, some set of images and sounds. Learners reuse theses software resources (programs and content) for themselves elaborately. For these reasons, we adopted Greenfoot as an environment and method of game software creation, but a single problem was still open: "how to teach".

C. *Project-based learning (PBL)*

Indeed some software is useful for FYE (first year education of universities), but outcomes of the education with use of such software depends on learning methods. As especially software is large-scale, complex, and changeable as mentioned before, appropriate teaching methods and theories should be selected accordingly. One of such good learning methods is PBL (Project-based learning) [10]. The root of PBL lies in American culture of education: experimental, hands-on, student-directed learning, i.e., "doing a project" is the best way to learn. Current PBL, which is originally based on Problem-based learning in the medical field, influenced by a revolution in learning theory partially based on neuroscience and psychology research, which have extended cognitive and behavioral models of learning. In the last decades PBL has been developing rapidly and now is applied to many educational fields. One of them is the field of software engineering education, and various good outcomes are reported. And many of them are concentrated on how to learn knowledge and skills well. While applying PBL to software education, we realized PLB is useful not only to learn knowledge and skills of the relevant subjects, but also to realize what should be learned to learn the subject adequately. From this point of view, we tried to examine if PBL is effective to holistic learning. This paper reports the results of our experience of the examination.

III. OVERVIEW OF OUR NEW PBL-BASED SOFTWARE LEARNING COURSE THROUGH GAME PRODUCING [11]

A. *Overview of Instructional design of our PBL course*

Our new course we propose is designed, according to the principle of tangible software education, to cover whole processes of software development. Details are as follows:

(1)Learning objectives: Learners figure out software development processes holistically by quasi-experiencing all processes through game software production. At the same time learners realize what kind of knowledge and skills should be acquired from the industrial point of view, and also understand why and what they have to study more from now on.

(2)Expected outcomes: Learners' awareness makes learning goals in the future more clear than before, and learners have more motivation to learn.

(3)Intended learners: University students with some Java Programming experience. Main target is student of second year at university. They are expected to know the basics of programming and programming language Java. Object-oriented programming is not expected to have been studied already.

(4)Learning style: In industry software is produced as a project by a groupware. For this reason, this course adopts the learning style of PBL (Project-Based Learning). At first, learners are divided into some groups of five members. Every group is regarded as a virtual company. Every member contributes to game software production as a virtual company's staff. They discuss and decide what game to make voluntarily and with responsibility. Everything is strictly decided for themselves.

(5)Teaching staff: A teacher and 8 TAs (Teaching Assistants) for about 80 learners. Each TA is a meta-project manager to manage 2 groups. In our course, a pair of TA manages 4 groups together.

(6)Teaching materials: A textbook is originally prepared for PBL by creating game software from upper process through lower process up to sales presentation. Details are in the next section.

(7)Learning environment: We adopted object-oriented programming language Java, because Java is one of the most important programming languages in industry. Moreover, Greenfoot is adopted as a programming development

environment, and we regard it as a learning environment, because learners can learn programming in a way of the "class first approach" as is proposed in BlueJ, where class first approach is a kind of learning method: "class design first, coding next." Every learner installs Greenfoot on their own personal computers all by themselves.

(8)Time schedule: The course we propose consists of 6 lessons. Learning term is 6 week long, and one day is 4.5 hour long.

1) First Day:

Preparing software developing environment by installing Greenfoot, Java SDK, drawing software GIMP, etc.

2) Second Day:

Learn how to use the software Greenfoot by creating some trivial game programs. In this phase, programming is not done in groups but individually. This phase is not PBL but self-learning as usual courses.

3) Third Day:

Virtual company is established by group to produce game software. This phase is PBL. Everything is decided by every team. Teaching staff gives only advice to virtual company as project mentor. They start to discuss what game to make, how to make it. Work schedule is also managed for themselves.

4) Fourth Day and Fifth Day:

Game production process goes on, while documents of planning, proposal, use case, software designs etc. are written as the occasion arises. When producing game software, many subtasks should be done, e.g., scenario writing, BGM music composing, picture drawing etc. They are assigned to virtual company staff according to their interests, qualification and readiness.

5) Last Day:

Game software is presented to other virtual company member not in technical terms but in commercial terms. After everything is over, all processes are reported by every virtual company with all documents written.

B. Design of Textbook

As a teaching material, a Textbook is prepared for learners to be accustomed to programming environment Greenfoot and to take PBL activity. The textbook starts with cover page, introduction, aims of the course, learning goals and scheme of the course. Then part I Preparing Greenfoot, part II trivial game production, and Part III game software creation project (PBL phase). Later comes references including URL and then samples of style files of many kinds of documents.

C. Examples of production

Here show some examples of learners' production is shown in [11]. In our course, learners are students of computer science at a technical university, so most games are shooting games, chasing game like packman (Figure 1) or adventure games like super Mario.

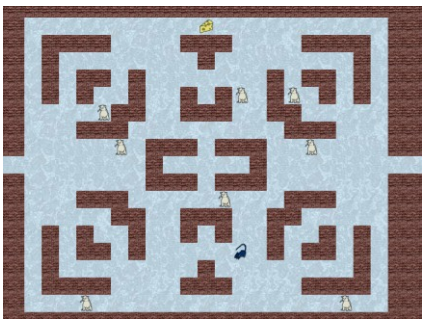

Figure 1. An example of games made by sutents, which is a chasing game that a mouse controlled by a human player goes for a piece of cheese, while cats are chasing the mouse to attack.

D. Result of evaluation in our first PBL course trial

To evaluate the fundamental validity of our proposal course, assessment was done by the questionnaire method. Figure 2 shows an evaluation result of attractiveness, which shows fundamental validity of the PLB course. More details are shown in [11].

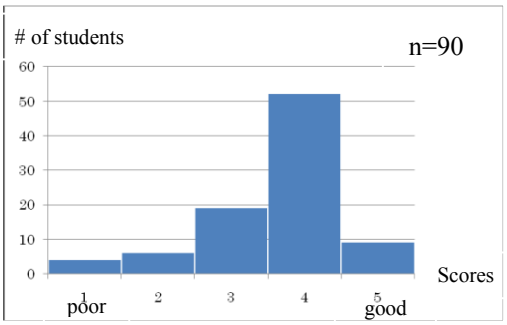

Figure 2. A Result of PBL in terms of Attractiveness.

E. Discussion

As the Figure 2 shows, our PBL course is so far so good, in a sense that most of students are satisfied with the PBL course. We also have an impression that students were willing to learn programming. On the other hand while we were giving the PBL course to university students, we have found students not only learned the content of the relevant subject, but also found out what should have been learned more to learn the subject more deeply at the same time. That is, our PBL course can make students learn the content and develop learners' self-awareness. So we planned to apply the PBL course to high school students, who in general have little amount of knowledge and skills of programming. From this point of view, we applied to our PBL course for high school students who participate in a summer science camp in this summer.

IV. NEW TRIAL IN A SUMMER SIENCE CAMP

A. Overview of Summer Science Camp

Every year in summer and winter, Japan Science Foundation gives a collection of experimental activities for

high school students of age 15-18 to learn scientific topics during a few days. This year we proposed a summer science camp course which was accepted by Japan Science Foundation. The course title was "Let's enjoy game software creation - Experimental introduction to software development."

B. Design of the course in the Summer Science Camp

The course was designed to give chances to high school students with little amount of programming knowledge and skills to learn what software development is. Of course we also planned to encourage students to go to IT fields in universities by showing attractiveness of software production. To accomplish this goal, we designed the PBL course to produce any game software which they wanted to make. PBL was adopted as a learning method. An outline of the course is presented in C.

C. Overview of Execution of the PBL course

1) Learning objectives:

Principally the same as in 3.1

2) Expected outcomes:

To give students the opportunity to learn what software development is, and what they should learn more in high school to study IT at university. Moreover, to attract students to go to IT fields in the future.

3) Intended learners:

High school students with little amount of programming knowledge and skills who have appetite for learning game software development. In this summer science camp, 20 high school students participated in the camp. Half was male students. Only two male students were familiar with programming, but others were novice programmers.

4) Learning style:

Same as in 3.1. But TA helped writing programs because students have poor knowledge and skills of game programming. In this sense, the principle "Everything is strictly decided by themselves" was violated partially. 20 students were grouped into five teams of four members each.

5) Teaching staff:

A single professor, two assistant professors, and two TA (Teaching Assistants). One TA was master candidate of the first year, other TA doctoral candidate of the third year. All staff is familiar with programming in Java, Greenfoot environment and PBL.

6) Teaching materials:

Adopted the same as in 3.2. But software installation was omitted and a supplemental lecture was given to let the students know what software development is and why they learn by a special learning method PBL. The Lectures took about 20 minutes.

7) Learning environment:

Same as in 3.1, but all software was installed by teaching staff.

8) Time schedule:

a) 1st Day

- 13:00～13:15 Opening ceremony
- 13:15～14:00 Introductory lecture

  To get accustomed to computer and software environment
- 14:00～15:00 Basic training (1)

  Learning how to use Greenfoot individually
- 15:00～15:50 Basic training (2)

  Learning how to use Greenfoot individually
- 15:50～16:00 Intermediate add-up
- 16:00 ～ 16:20 Lecture to know what software development processes more deeply

  Title: Why software is important in our society?, and what people are needed in IT industry?
- 16:20～17:00 Explanation of PBL and establishment of virtual companies

  Organization of a team as a virtual company, assigning roles to team staff, and preparation for the following day's activities
- 17:00～18:30 Welcome party
- 18:30～19:00 Move to Hotel
- 19:00～22:00 Meeting to make friends

b) 2nd Day

- 9:00～ 9:20 Explanation of the day's activities and confirmation of what they had learned on the preceeding day
- 9:20～10:00 Game producing (Game planning and writing a draft of plan document)
- 10:00～11:00 Game producing (finish writing plan document)
- 11:00 ～ 12:00 Game producing (specification document, external design)
- 12:00～13:30 Lunch
- 13:30～14:30 Game producing (resource producing: drawing image and composing music etc.)
- 14:30～17:00 Game producing (coding and etc.)
- 17:00～17:30 Move to Hotel
- 18:00～19:00 Dinner
- 19:00～22:00 Meeting

c) Last Day

- 9:00～11:00 Game producing (Level design and etc.)

- 11:00～12:00 Preparation of presentation
- 13:30 ～ 15:00 Presentation with game demonstration
- 15:00～16:00 Playing Games
- 16:00～16:40 General add-up

  What we have learned so far and what should we learn more from now on
- 16:40～17:00 Awarding and closing ceremony

D. Outcome of the PBL course

1) Products as outcomes

High school students could manage to produce games, such as shooting games, obstacle-avoiding game, typing game, and goods searching games. Only typing games are well produced. Students seem to be short of time to produce other types. Probably more time should be allotted for these activities in the future.

E. Evaluation

To evaluate the fundamental validity of our proposal course, assessment was done by the questionnaire method as described below:

(1) Aims: To investigate the fundamental validity of the proposed course mainly in terms of what learners can learn.

(2) Subjects: 20 students (20 out of 20 answered.)

(3) Method: questionnaire method with 18 items. Questions are as follows;

Q1. Level of learning content (m=4.3, σ=0.9)
Q2. Interest of learning content (m=4.8, σ=0.4)
Q3. Freshness of content (m=4.8, σ=0.4)
Q4. Teaching materials are well described (m=3.6, σ=0.7)
Q5. Satisfaction of teaching materials (m=3.7, σ=0.9)
Q6. Amount of explanation time (m=3.0, σ=0.6)
Q7. Amount of work time (m=2.0, σ=0.9)
Q8. Amounts of work (m=3.3, σ=1.3)
Q9. Total number of work days (m=1.7, σ=0.7)
Q10. Quality of teachers' teaching ability (m=4.5, σ=0.9)
Q11. Quality of TAs' teaching ability (m=4.4, σ=0.9)
Q12. Classroom (m=4.1, σ=0.9)
Q13. Overall satisfaction (m=4.7, σ=0.5)
Q14. Comments (free description)
Q15. What can you learn? (free description)
Q16. What do you know about yourself? (free description)
Q17. Learner's effort (m=4.2, σ=0.9)
Q18. Accomplishment level (m=,3.8 σ=0.8)
Q19. Programming level of yourself (m=1.4, σ=0.8), where score ranges from 1 to 5, i.e., from very poor, poor, so so, good, very good, respectively.

(4) Results: In general, the content in the summer science camp was difficult for high school students, but most students learned what software development processes are and also what they should learn more at high school to study IT at university.

Questionnaire method shows PBL can be also useful to learn not only content of subject, but also to aware of what should be learned more/better, e.g.,

a) I could take a role in my team of even first-met friends.
b) I have found leadership and making friends very important.
c) I have to learn mathematics and English more.
d) I am poor at PC, so I have to try to learn PC more.
e) Programming is easier than I thought ever.
f) Other people did not dislike me so much.
g) I have ability of writing document.
h) I found I am shy, I have to conquer it.
i) I could express myself, etc.

V. CONCLUSIONS

In this paper, we proposed and discussed a new software learning method using PBL through game production. PBL is generally regarded as one of outstanding learning methods, on the other hand, we pointed out PBL is also useful for learners to learn not only learning materials but also to develop learners' self-awareness, e.g., *why do we learn the content?*, *what should we know further more?* and so on.

ACKNOWLEDGMENT

We thank Tangible Software Engineering Education Project staff and students of Thought and Language laboratory of Tokyo University of Technology, especially Hiroshi Maruyama of doctoral candidates in the 3rd year.

REFERENCES

[1] T. Nakamura, "What's Tangible Software Engineering Education?," 1st Int. Symp. Tangible Software Engineering Education, STANS09, Tokyo, 2009, pp.1-8.

[2] D. L. Kirkpatrick and J. D. Kirkpatrick, *Evaluating Training Programs: The Four Leves*, Berrett-Koehler, 2006.

[3] J. L. Ford, *Scratch Programming for Teens*, Course Technology, 2008.

[4] W. P. Dann et al., Learning to Program with Alice, Pearson Prentice Hall, NJ, 2008.

[5] Robocode. Available: http://robocode.sourceforge.net/

[6] D. J. Barnes and M. Kölling, *Objects First With Java: A Practical Introduction Using BlueJ*, Prentice Hall, 2008.

[7] P. Henriksen, *A Direct Interaction Tool for Software Engineering Education*, M.S. Thesis, University of Southern Denmark, 2004.

[8] M. Kölling, *Introduction to Programming with Greenfoot*, Prentice Hall, 2010.

[9] Reiner Consulting, Kolb-Learning Cycle, Available: http://www.reinierconsulting.com/kolb.html.

[10] Project Based Learning, 2nd ed., BIE, CA, 2003.

[11] H. Kameda, "PBL-based first year education course of learning software engineering for novice software programmers through game software production," in Proc. IADIS2010, e-learning2010, Freiburg, 2010.