# Effect of Programming Education using Software Development Methods

**Yasuyo KOFUNE**
**Osaka Prefectural Yodogawa Technology High School**
**Osaka, Japan**

**and**

**Takahiro KOITA**
**Doshisha University**
**Kyoto, Japan**

**ABSTRACT**

It is necessary for high school students to learn knowledge and skills of software design in order to improve their programming skill. Students participate in a programming contest in class as a way to improve programming skill while enjoying programming. In programming contest application creation, Agile, MindMap, UML and pair programming techniques are used. In this study, we show the effect and problem of improvement of students' programming skill by using these methods, and consider the student software design class method in which high school students cooperate.

**Keywords**: Programming Education, Agile Development, Mind Map, UML, Pair Programming

## 1. INTRODUCTION

Many Japanese technical high school students work as manufacturing engineers after graduation. To work as engineers, not only expert knowledge and skills, but also logical thinking ability, problem solving ability, and imagination are required. Therefore, Japanese technical high schools have programming classes for first-grade students to learn not only the knowledge and skills of programming but also logical-thinking ability and problem-solving ability. In programming classes, students learn programming language grammar and algorithms of basic processing such as loop processing and selection processing by repeating inputting and executing sample programs from a textbook, when they finished the textbook, they can practice to create applications[1]. Students enjoy programming and learn voluntarily in the early stages of programming learning. However, as programming learning progresses, some students quit. There are various reasons for students giving up on programming learning. For example, students who are not good at typing cannot proceed smoothly because much time is spent looking for errors due to typing mistakes[2]. Some students might have no interest in the result of running the sample program. Students who have expectations in programming classes are dissatisfied with not being able to write programs of their own such as games. In order for students to have fun learning programming, there is an approach to upgrade student motivation and reduce typing and grammar learning, using visual programming languages like Scratch[3].

We are considering a class method aimed at students enjoying programming and improving programming skills.

Our study goals are as follows.

- Students enjoy programming by creating applications with their own ideas.
- Students gain confidence in programming by completing the application.
- Students want to learn more about programming and create applications again.
- Students improve their ability to think logically by programming their own ideas.
- Students improve problem-solving skills through trial and error in the course of application creation.
- Students cooperate with other students to improve imagination, expressive ability, and communication skills.

As a way to achieve the above objectives, we designed a programming lesson using a programming contest. Experiences of students making applications with student's own ideas will be knowing the enjoyment of programming, confident in programming lead to improved programming skills.

Students who participate in the programming contest have a clear goal of completing their application. This goal leads to voluntarily learning programming. Completion of the application makes it possible for students to realize that programming learning until this time is progressing well. With the completion of the application, the students can see the result of the programming learning until this time. Students are motivated to learn programming in the future by seeing the results. Voluntary programming learning leads to improvement of student programming skills.

Students who participated in the programming contest used Agile[4], MindMap[5][6], UML[7], and pair programming methods in application creation. In this study, we show the effects and problems of improvement of students' programming skills using these methods, and show the necessity for students to learn software design.

## 2. OUR APPROACH

The programming contest in which the students participated was a high school PC contest (mobile department)[8]. This contest was for students to create and present Android applications by team (three members) according to a predetermined theme. Such contests with students using their own ideas can help them enjoy programming and improve their programming skill. Students who participated in the contest were not good at programming and had no experience creating Android applications. However, students actively created using MindMap, UML and pair programming on the basis of the Agile method.

The following describes the aim of using Agile, MindMap, UML and pair programming methods for creation of an Android application.

**Agile:**
This cooperative development of software by the students demonstrates the effectiveness of Agile Development.Agile Development is an umbrella term for several iterativeand incremental software development methodologies.

The following two points illustrate the Agile Manifesto's main principles [4]:

- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.

Based on the concept of Agile, which emphasizes dialogue, students make Android applications through dialogue. Students collaborate on all Android application creation. In order to create better applications, it is important to allow other members' weak points and to mutually acknowledge each other. By talking each other, the students share ideas of each other and understand what other members are doing. Application creation using Agile leads to improved cooperativeness and communication skills of the students. Also, students have the awareness that they are participating in the creation and have confidence that there is something they can do.

**MindMap:**
Students used MindMap to organize ideas for Android applications. MindMap draws a concept (theme) in the center of a piece of paper using keywords and images, and connects keywords and images radially associating from there, and expands the idea. Students who are not good at putting together ideas can use MindMap to organize thoughts and expand their thoughts. MindMap also makes it easy for multiple students to add ideas, share ideas, and collaborate. Figure 1 shows an example of the Mind Map.

**UML:**
Students used UML to grasp the overall structure of the application. Students had never used UML before, so they first learned about UML. The students seemed to understand a little of the use case diagram, activity diagram, sequence diagram, but class diagrams and object diagrams were difficult. Students gave up using UML and shared information on the functions and screen design of Android applications in Japanese and with illustrations. Figure 2 shows an example of the UML class diagram.

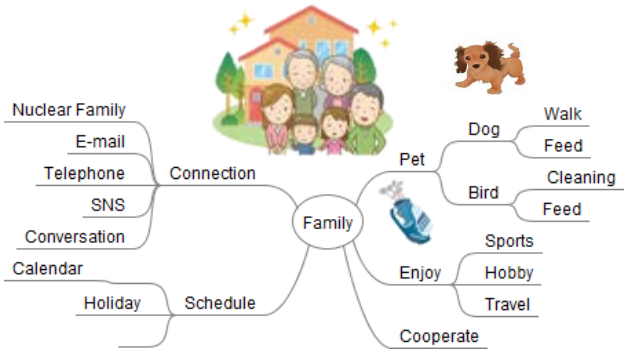


Figure 1: Example of Mind Map

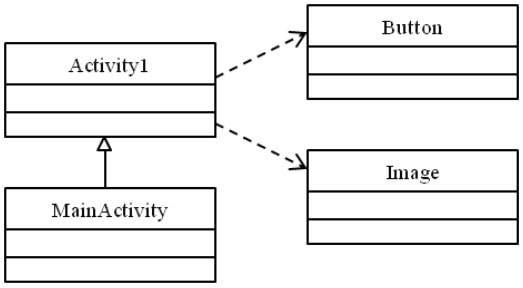


Figure 2: Example of UML class diagram

**Pair programming:**
Students participating in the contest differed in programming ability and typing skill. In pair programming, while students can complement each other's missing abilities, they can cooperate to create programs[9]-[11]. Students created the program in two ways. One was to write code using Java language, and the other was to use AppInventor. AppInventor is a programming tool that combines blocks with built-in processing. Using AppInventor does not require typing code[12].

Programming in the Java language requires students to learn the grammar of the Java language and how to use the development tools, but because the AppInventor tool is intuitively usable, students can start using it immediately.

Programming with Java language is difficult for students. When a problem occurs in the program, the students explore the cause of the problem and think about the solution. In programming with the Java language, the students often have to explore whether the cause of the problem is processing logic, a grammar mistake, a typing error or the like. Students take time to identify the cause of the problem and it takes more time to solve further.

In programming with AppInventor, students are able to narrow down the cause of the problem to the logic of processing, and it is easy to find a solution through trial and error.

## 3. DISCUSSION

The following describes the effects and problems of Agile, MindMap, UML and pair programming methods.

**Agile:**
Based on the concept of Agile, students cooperatively created Android applications. As students talk and create Android apps, they can clarify what they did and also understand what other students are working on. Students have awareness that they themselves are participating in the creation. As students' dialogue increased, the creation of Android applications proceeded more smoothly. The cooperative work of the students supplemented not only information sharing but also immature technological understanding. Also, when the creation did not go smoothly, the students were able to work without giving up [13][14]. Conversation among students helped greatly in creating applications in cooperation. By talking, the students could learn about each other 's good and weak points. As the students knew what they were doing, even if someone was late, they could support smoothly. Moreover, they were able to decide to share the work with considering the appropriateness of each individual without misunderstanding. The students were able to create applications fun by enjoying good cooperation among students towards the common goal of application completion. Students had a sense of accomplishment with the completion of the application and gained confidence in their abilities, which led to motivation to continue programming learning.

**MindMap:**
Students initially came up with no ideas as to what kind of Android application to create. Students, when trying to summarize their thoughts in sentences, thought that it was necessary to have proper sentences. It took too long for the students to express their thoughts in sentences and to share it with other students. By using MindMap, students became to discuss various ideas in a short time and all the students were able to participate in the discussion. MindMap connects words and illustrations, so all the students were able to draw. MindMap made it easy for multiple students to create, organize and add ideas.

**UML:**
The students who participated in the programming contest did not have experience writing programs or creating applications based on their own ideas. They seemed to be completely unaware of what they needed and what they decided and how to process them in order to make their ideas an application. Just learning grammar in the programming language, students did not know how to think about how to handle their ideas by computer.

Students tried to use UML to share information on the functions and structures of Android applications, but they did not have UML knowledge and it was too complicated to understand. Students used Japanese and diagrams to organize the functions required for the application and think about the processing method. However, even with Japanese and figures, the students could not smoothly consider the processing method. The reason was that students had no experience creating applications, and lacked the ability to think logically, express their own ideas, and convey their thoughts to others. It is necessary for students to put together their own ideas and learn to think about the mechanism to achieve this. To write UML, the students need to understand the required specifications and be able to think logically. Learning UML leads to improvement of expressiveness and logical thinking skills of students. It is also effective to give students the ability to create applications with their own ideas. Learning UML is extremely beneficial [15][16].

**Pair programming:**
Pair programming did not work for programming in the Java language, but it worked well with programming with AppInventor. In creation using the Java language, because there were no students with high programming skills, none could proceed smoothly. In creation using the Java language, because there was no student with high programming ability, no one was able to lead other students and creation could not proceed smoothly. However, in creation using AppInventor, students were progressing well while interacting with each other. AppInventor takes time and effort to prepare a program shared by two or more people, to connect the created program and to complete the application. A program created by AppInventor can't exchange code easily. AppInventor is not suitable for application creation by multiple people. However, pair programming, which creates one screen together, is suitable for programming with AppInventor.

Next, characteristic problems in the case of creation by AppInventor will be described. AppInventor creates screen design and application processing by combining blocks with processing incorporated. For Android application creation, using the blocks provided by the AppInventor tool, there is a limit on the size of the application that can be created. Due to that constraint, there were cases where a major change was necessary when programming student ideas. Also, there were occasions when it took more time and effort, more useless processing than programming with the Java language. In creation using AppInventor, students sometimes started programming without thinking about processing method beforehand. They thought that as a result of rough combining the blocks, they were lucky to be able to do it. When students create without thinking about the design of the application, it takes a lot of work to add information sharing about the design of the application and add functions. Application creation without considering application design has little effect on students' ability to think logically. For students to be able to write programs with their own ideas, it is also important to improve their logical thinking ability.

Programming with AppInventor is a motivator for programming learning by students who are not good at programming and students who give up programming learning. However, it is not enough for students who work as engineers in the future to acquire knowledge and skills of software design, to improve programming skills, logical thinking ability and problem solving ability. It is not enough to study grammar in programming languages; learning software design is necessary to learn programming skills that help students think logically so they can write programs from scratch. Learning contents of programming classes are grammar of programming language, basic algorithms, requirement analysis in program development, software design and testing technique. However, many of the students will not be able to keep up with the classes in grammar learning in programming languages before learning the requirement analysis, software design and testing techniques in program development. Many of the students are losing interest in grammar learning in programming languages and are giving

up because programming is difficult. Therefore, in programming classes, students are not advanced enough to learn software design. Learning programming language grammar and software design at the same time in programming lessons is a heavy burden for the students. Learning of software design needs to be done separately from learning the grammar of programming language. We examine a method of teaching software design in a class of practical training, which students can consider and output by themselves.

## 4. CONCLUSIONS

From experiences participating in the programming contest, students were required to master the knowledge and skills of software design in order to improve their programming skills satisfactorily. In the future, we are planning to propose a class method of learning software design in which technical high school students cooperated with each other, and to verify its effectiveness.

## 5. REFERENCES

[1] Japanese government guidelines for high school education (Information subject area), Ministry of Education, Culture, Sports, Science and Technology-Japan (MEXT), 2010.

[2] S. Dohi and N. Konno, "Using the Processing for Practice of Introduction to Computer Programming Education for High School Students," Information Processing Society of Japan,Information education symposium (SSS2013), Vol.2013, No.2, pp. 217-224, 2013.

[3] Branko Kaucic and Teja Asic, "Improving introductory programming with Scratch?," MIPRO, 2011 Proceedings of the 34th International Convention, Opatija, Croatia, pp. 23-27, 2011.

[4] Manifesto for Agile Software Development, http://www.agilemanifesto.org/.

[5] Tony Buzan - Inventor of Mind Mapping, http://www.tonybuzan.com/about/mind-mapping/.

[6] T. Buzan, Mind Map Handbook: The Ultimate Thinking Tool, Thorsons, 2005.

[7] UML: Object Management Group, http://www.uml.org/.

[8] High School PC Contest, http://www.u-aizu.ac.jp/pc-concours/.

[9] A. Cockburn and L. Williams, "The Costs and Benefits of Pair Programming," Extreme Programming Examined, 2000.

[10] L. Williams and R.L. Upchurch," In Support of Student Pair-programming," Proc. of the SIGCSE Technical Symposium on Computer Science Education, pp. 327-331, 2001.

[11] Tie Hui Hui and Irfan Naufal Umar, "Pair Programming and LSs in Computing Education: It's Impact on Students' Performances ", 2011.

[12] MIT App Inventor, http://appinventor.mit.edu/explore/.

[13] Vladan Devedzic and Sas˘a R. Milenkovic, "Teaching Agile Software Development: A Case Study," IEEE Transactions on Education, Vol. 54, Issue 2, pp. 273-278, 2011.

[14] Marcello Missiroli, Daniel Russo and Paolo Ciancarini, "Learning Agile software development in high school: an investigation," ICSE '16 Proceedings of the 38th International Conference on Software Engineering Companion, pp. 293-302, 2016.

[15] I. Diethelm, L. Geiger, and A. Zündorf. "Teaching Modeling with Objects First," Proc. of the World Conference on Computers in Education, 2005.

[16] C. Starrett, "Teaching UML Modeling Before Programming at the High School Level," Proc. of the IEEE International Conference on Advanced Learning Technologies, pp. 713-714, 2007.

Relevant Software offers established practices of software product development. Following them is a driving factor for the successful delivery of solutions.Â  When digging deeper, software product development is a highly organized process with precise procedures and strictly defined steps known as Software Development Life Cycle (SDLC). Whenever you need a sophisticated system, software suite or end-user web or mobile app your outstanding project delivery, besides all the other important factors, largely depends on a set of processes practiced by the development team. For software developers, understanding the software development life cycle (SDLC) facilitates the effective planning and delivery of high-quality software products. This article takes you through the software development life cycle, providing an overview of the process and limitations of various implementations of the SDLC.Â  Developers and software engineers use it to create effective plans and designs. They also apply the various SDLC stages to develop innovative software products.Â  Various programming languages can be used. These may include languagues such as PHP, Java, C++, Pascal, C, etc. The language chosen depends on the type of software being developed, the business use-case among other constraints. Testing. With the fourth industrial revolution, software education has become much more important. In South Korea, Informatics will be taught as mandatory to middle school students; education program development and applications with block based programming languages are vivaciously conducted.Â  swKim & yjLee, "Development of a Software Education Curriculum for Secondary Schools," Journal of the Korea society of computer and information, Vol. 21, No. 8, pp. 127-141, 2016. Design a Programming Education Plan for SW Education Using Robot and Mobile Application Development Tool. Jan 2014. 615-624.Â  Comparison of the Effects of Robotics Education to Programming Education Using Meta-Analysis. Jan 2014. 413-422.