International Journal of Computer Science and Software Engineering (IJCSSE), Volume 7, Issue 1, January 2018
ISSN (Online): 2409-4285 www.IJCSSE.org Page: 12-16

A Development of Static Analysis Program and Its Application

Wanwipa Titthasiri, Asst.Prof.,Dr.[1] and Chantapat Tancharoen[2]

[1, 2] Department of Computer Science, Rangsit University, Pathumthani, 12000, Thailand

[1]awanwipat@gmail.com, [2]chantapat.ct@gmail.com

ABSTRACT
Thailand is moving forward to 'Thailand 4.0' which is an economic model aiming to promote creativity, innovation, and technology for developing the country. Therefore, it is necessary to produce high-quality graduates in order to invent those innovations and technology. The main purpose of this research was to develop a static analysis program to evaluate programming behavior of first-year students who enrolled an introductory programming course. The knowledge of compiler and software engineering were used to develop this program. It evaluates student's programs in software metric as an indicator to represent different programming behaviors. In addition, it also produces statistical reports and graphs. Finally, the research result shows the strengths and weaknesses of students' programming capability. It is believed that this finding would be a valuable guideline for lecturers to redesign his or her basic programming course to improve the students programming capability.

Keywords: *Static Analysis, Analysis Program, Programming Teaching, Novice Programmer, Computer Science.*

1. INTRODUCTION

"While hardware is now the largest single IT budget item for most small and medium businesses (SMBs), it will be the slowest growing category over the next four years for that group".[2] Moreover, Darrow [2] pointed that as of this year more than half of the money will be spent on software. It was shown that spending by SMBs on hardware was 1.1% compound annual growth rate (CAGR) over the next four years compared to 6.6% growth in software spending. Consequently, employment opportunities for computer science (CS) graduates are strong growth in the coming years. SIPA (Software Industry Promotion Agency)-Public organization in Thailand-reveals software market survey 2016, software production value grows 4.3%.[5] It was reported that 42.6% of all ICT employment category in 2016 was software design and programming.[5] Furthermore, AEC (Asian Economic Community) and the policy of Thailand 4.0 are coming in a few years. Thailand may encounter a crisis. But if Thai CS graduates are well prepared thoroughly, the AEC would not affect them badly.

Despite the fact that there is an increasing demand for CS graduates who will be experts in developing software and its innovation, the programming skills of recent CS graduates typically fall short of employers' expectations.[5] In addition, one of the important reasons for rejecting CS major study, were its' difficulties, especially in programming and technical courses,[12] Research has shown that first year CS students struggle with learning how to program and understanding programming concepts.[7] Even though, CS education is not the study of programming, but programming is an important tool for the student in CS learning and working in ICT realm. It is a core skill for first year CS students. However, it is found that students' programs are often poorly constructed because beginning students often try to solve a problem as quickly as possible without thinking about the quality of their programs.[10] Learning to program is a difficult process. In order to become a highly competent in programming, student programming exercises should be needed to analyze and find out their weaknesses. These could be considered as alternative solutions in programming class to improve student performance in programming courses.

2. OBJECTIVES

The objectives of this study are: 1) to develop a static analysis program; 2) to apply this program for doing an analysis of student programming exercises; and 3) to assist lecturers to improve their beginning students 'programming teaching.

3. LITERATURE REVIEWS

3.1 The Program Analysis

The program analysis has been classified into two types of analyses: static and dynamic. Static analysis is the process of examining source code without executing the program. Dynamic analysis is the process of running a program through a set of data.[10] The literature indicate that there are some systems as program analysis to help novice students learn to program. For example: Talus[11] is an automatic program debugging tool for the Lisp language; 2) CourseMaster[4] is a client-server system providing functions for automatic assessment of students work in Java and C++; 3) Expresso [6] is designed to identify beginning student Java programming errors; 4) ELP[8] is an online interactive and constructive environment for learning to program. Much research has used these systems to help novice students to learn to program. Most of them are large systems and are designed for Java, C++ source codes. In this study, a static analysis program was developed to help lecturers learn about beginning students' programs. It is used to show problems in C# code including unnecessary complexity and software metric which is a way to measure the quality of programs.

3.2 Previous Works

Truong, Roe, and Bancroft[10] introduced a static analysis framework which can be used to give beginning students practice in writing better quality Java programs and to assist teaching staff in the marking process.
Taherkhani, Malni, and Korhonen[1] addressed the problem of automatic algorithm recognition and introduce a method based on static analysis to recognize algorithm.
Truong, Roe, and Bancroft[9] described a "Fill in the gap" program analysis framework which tests students' solutions and give feedback on their correctness, detects logic errors and provide hints on how to fix these errors.
Chen and Lin [3] examined correlations between students' learning styles and their performance in writing and debugging Java programs.
Toomey [13] reported on a tool called Arjen which is designed to quantify the errors made by novice programmers.
In the literature, existing research has used program analysis system as a tool to give students write program and useful feedback. By these tools, people learn novice programmers' errors, algorithms, and how to fix them. But no research has been done to evaluate software engineering metrics and to determine how the constructs of student's program compare with standard or model program, written by the expert. Much research has shown that beginning students' programs are often poorly constructed. Therefore, this paper introduces a static analysis program which uses both software engineering metrics and relative comparison to judge the quality of beginning student's program and finally provide suggestions to lecturers about how is the programming teaching for the first year CS student improved.

4. METHODOLOGY

The approach in this study is based on both software engineering and compiler construction. The researchers divided the study into the following two phases: a static analysis program's development and its application.

4.1 A Development of Static Analysis Program

There are many techniques to develop static analysis program. Lexical analysis and syntax analysis have been adopted in compiler construction application vary from string matching based on the program source code to matching program paragraph representations. Software metrics are the results of this static analysis program. They have been used to evaluate beginning student's program. For example, line of codes, number of local variables and global variables, Halstead software metrics, and so on. Moreover, some techniques of optimization phase in compiler construction are also adopted in this program because they provide useful information about the structure of a program which measure the difficulty or complexity of program.

4.2 Program Implementation

1) *Participants* This study was participated by 23 CS freshmen who enrolled in an introductory C# programming course offered by the department of computer science at Rangsit University, Thailand. All of them had not had programming experience.
2) *Procedure* The experiment was conducted in the second semester, 2016. All participants were offered lectures and hands-on exercises which took place in a computer lab. There were two 120-minutes class periods per week. Programming constructs covered included, input/output, variables, arithmetic and logic expressions, classes, methods, conditional statements, loops, and arrays. The following three problems, shown in Fig.1 were intended to assess students' comprehension of programming concepts, their ability to apply these concepts in programming tasks-problem solving, algorithmic thought, and C# code transforming. In this study,

students were asked to write all of three complete C# programs in the hands-on programming tests.

Problem #1
Write a complete C# program to input 2 integers and print out all of integers between 2 inputs by ascending sort.
Problem #2
Write a complete C# program to input 2 integers and make the division. The less number is divisor. Division result and its remainder are printed out.
Problem #3
Write a complete C# program to compute the area of triangle, square, and circle using methods. Users can choose which figure they want. For example, if user chooses to compute the area of circle, then program will ask a number of radius for computing its area.

Fig. 1. Three Problems for Hands-on Programming Tests.

3) *Data Collection* All of the hands-on programming test problems, each participant was asked to write three complete C# programs without any errors. After then, all of them were submitted and stored in the "StaticAnalysisDB" folder on the server and only loaded when they are required analyses.

5. RESULTS AND GUIDELINES

5.1 Software Engineering Metrics

For each of the hands-on programming test problems, the static analysis program gave some software engineering metrics, as shown in Table 1.

Table 1: Software Engineering Metrics

Software metrics	Meaning
Nom	Count the total number of methods
NoP	Count the total number of parameters
NoGV	Count the total number of global variables
NoLV	Count the total number of Local variables
NoAS	Count the total number of assignment statements
NoS	Count the total number of statements
NoB	Count the total number of blocks
NoL	Count the total number of loops
LOC	Count the total line of codes
Difficulty	Halstead complexity metrics

5.2 Experimental Results

The purpose of this analysis is to evaluate how the student's program compares with standard or model program. In the analysis, Euclidean distance was used to compute the difference among them, as shown in equation (1)

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(pi - qi)2} \qquad (1)$$

For each of the hands-on programming test problems, all of software metrics of each student's program were plotted in line graphs, classified into four lines, as following:
Average line means the average of software metrics of all students' programs.
Standard line means software metrics of standard or model's program.
Best line means software metrics of student's program which its Euclidean distance is the smallest value compared to standard's program.
Worst line means software metrics of students' program which its Euclidean distance is the biggest value compared to standard's program.
By comparing students' programs with standard program, four line graphs of each programming test problem were presented in Fig. 2, 3, and 4, respectively.

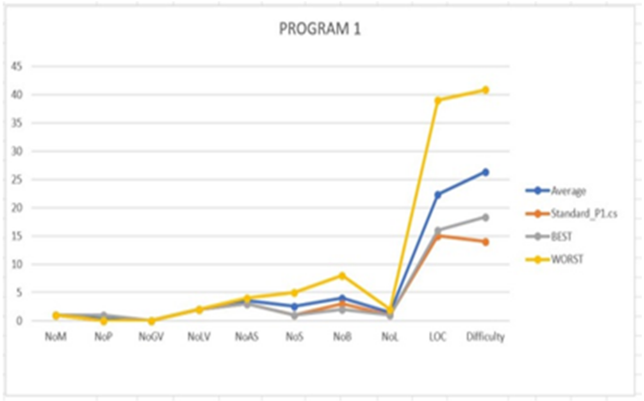

Fig. 2. A Comparison of Software Metrics for Program #1.

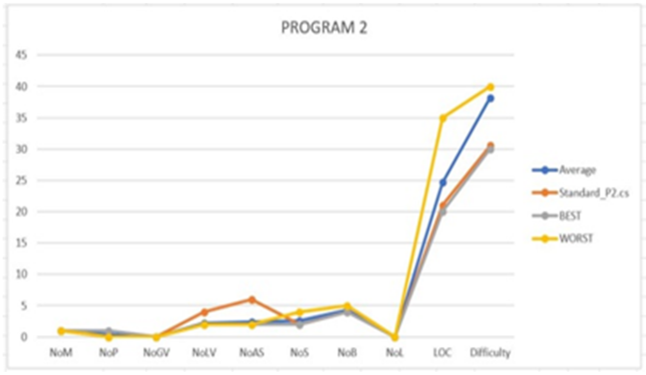

Fig. 3. A Comparison of Software Metrics for Program #2

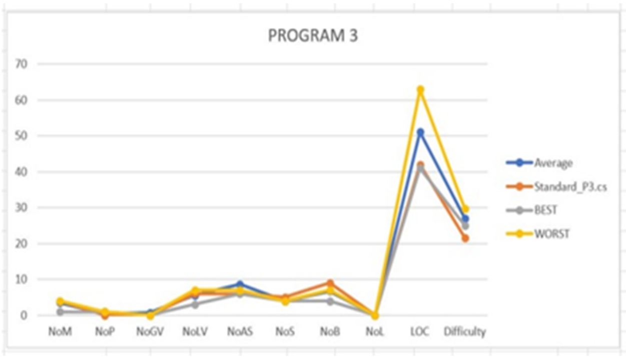

Fig. 4. A Comparison of Software Metrics for Program #3

As shown in the above graphs, all of three graphs (programming test problems) indicated the similarity directions. The significant unmatched points between students' programs and standard program were LOC and the difficulty. Both of LOC and difficulty values of students' programs were higher than values of standard programs. This can be used to predict the student's performance in writing program. High value of LOC and difficulty have shown that transforming from problems to algorithm were complex and selecting C# instructions to represent algorithm were not good enough to make a short and worth program. It may encounter readability and writability crisis which are criteria of the evaluation of software quality.

5.3 Guidelines

An analysis conducted in this study was to look at specific CS students' group as determined by software engineering metrics. Students' programs and standard program were compared to determine the difference of program writing. Specifically, LOC and the difficulty that students' programs obtained were significantly high compared with the standard program. This shown that problem solving and understanding in C# instructions were difficult for students in programming learning. Therefore, some suggestions would be proposed for lecturers to improve programming teaching in a beginning programming class, as shown in Table 2.

Table 2: Guidelines to improve programming teaching

Actions	Objectives
Assign more and more hands-on programming problems. (both in computer lab and homework)	More practices will get more experiences. Students will learn to solve problems and create algorithms.
Give solution as model for all assignments immediately.	Students will learn how good program is and will be able to apply them in the next practices.
Set up a small class.	With large class size, it is difficult for lecturer to synchronize his/her heavy schedules to provide additional help when the students need it.[9]
Give feedback on student's errors.	To provide hints on how to fix these errors.
Give students follow the coding standards.	First start learning in software engineering practices.

6. CONCLUSIONS AND FUTURE WORKS

In this study, the researchers developed a static analysis program for software engineering metrics as same as tools like CourseMaster and so on. This program extends the existing work in a comparison between students' programs and standard program. Due to time constraints, the researchers wrote a static analysis program to receive and analyze only a single C# source code file at a time. Therefore, this program needs to be extended or rewritten so a complete, multi-file C# program can be analyzed. Moreover, java program or C++ code should be able to be also analyzed. However, this study also offers useful suggestions as a guideline for lecturers who teach beginning CS students to improve teaching to program. It is expected that these suggestions would be useful and helpful for lecturers or teaching staffs in programming classes to place stronger programming skills into CS graduates who will be a valuable human resource to develop our country in the future.

REFERENCES

[1] A. Taherkhani, L. Malmi, and A. Korhonen, “Algorithm Recognition by Static Analysis and Its Application in Students’ Submissions Assessment,” Proc. Of the 8th International Conf. on Computing Education Research, Koli, Finland, pp. 88-91, 2008.

[2] B. Darrow, “Here’s More Bad News for Tech Hardware Makers”, FORTUNE Data Store, November 28. 2016. Available at: “www.fortune.com”

[3] C. Chen and J. M. Lin, “Learning Style and Student Performance in Java Programming Courses,”. Available at: “www.worldcomp- proceedings.com/proc/p2011/ FEC3196.pdf”

[4] CourseMaster:School of Computer Science and IT, The university of Nottingham,UK. Availableat: http://www.cs.nott.ac.uk/CourseMaster/cm_com/index.html Accessed2002/”

[5] Digital Economy Promotion Agency, Under the Administratrive Supervision of the Minister of Digital Economy and Society, 2016. Available at: “http://www.sipa.or.th”

[6] M. Hristova, A. Misra, M. Rutter, and R. Mercuri, “Identifying and Correcting Java Programming Errors for Introductory Computer Science Students,” Proc. Of the 34th SIGCSE Technical Symposium on Computer Science Education, Reno, Nevada, USA, vol. 34, pp. 153–156, 2003.

[7] N. Truong, “A Web-Based Programming Environment for Novice Programmers,” Dissertation, Queensland University:Australia, July, 2007. Available at: “www.eprint.qut.edu.au/16471/1/Nghi_Truong_Thesis.pdf”

[8] N. Truong, P. Bancroft, and P. Roe, “A Web Based Environment for Learning to Program,” Proc. Of the 26th
Australian Computer Science Conf., Adelaide, pp. 255-264, 2003.

[9] N. Truong, P.Roe, and P. Bancroft, “Automated Feedback for ‘Fill in the Gap’ Programming Exercises,” Proc. Of the 7th Australian Conf. on computing Education, vol. 42. NewCastle: Australia, pp.117–126,2005.

[10] N. Truong, P.Roe, and P. Bancroft, “Static Analysis of Students’ Java Programs,” Proc. Of the 6th Australian Conf. on Computing Education, Dunedin: New Zealand, pp. 317–325, 2004.

[11] W. M. Murray, Automatic Program Debugging for Intelligent Book Tutoring Systems, Morgan Kaufmann, Pittman, London,1989.

[12] W. Titthasiri, “Efforts to Revitalize the Computer Science Education in thailand,” International Journal of Engineering research and Applications (IJERA), Vol. 6, Issue 2, Feb. 2016, pp. 34-41. Available at: “www.ijera.com”

[13] W. Toomey, “Quantifying the Incidence of Novice Programmers’ Errors,”.Available at : “minnie.tuhs.org/Programs/BlueJErororslarjen_draft.pdf”