A Development of Computer Programming Analyzer: A Case Study on PHP Programming Language

Suchot Sinthsirimana and PatcharinPanjaburee
Institute for Innovative Learning, Mahidol University, Thailand
Email: suchot.sin@student.mahidol.ac.th, panjaburee_p@hotmail.com

***Abstract*—Competency in computer programming is important not only for the benefit of organizations, industry, and technology, but also for undergraduate students' performance at higher levels of computer science and, more generally, for software engineers. Due to the high importance of computer programming, the strengths and weaknesses of undergraduate students in this subject need to be identified. Weaknesses in particular need to be exposed and remedied to avoid carrying on students' shortcomings to higher school levels or worse in the working environment as professionals. Consequently, this study developed an innovative system named Computer Programming Analyzer (CPA) to evaluate the development and performance of students during computer programming courses. The results showed that the students who participated in the guidance from the CPA could improve their computer programming ability. Moreover, the CPA could be used as a supplement tool for teachers or project leaders to check the progression in computer programming.**

***Index Terms*—computer programming, software analyzer, diagnostic assessment, software engineering**

I. Background and Motivation

Most students or programmers gain their programming expertise by extensive exercise. Competency in computer programming is important not only for the benefit of organizations, industry, and technology, but also for undergraduate students' performance at higher levels of computer science and, more generally, for software engineers. Due to the importance of computer programming, the strengths and weaknesses of undergraduate students in this subject need to be identified. Weaknesses in particular need to be exposed and remedied to avoid carrying on students' shortcomings to higher school levels or worse into the working environment as professionals [1].This implies that proper learning guidance is required to improve student performance in computer science and education [2]. Generally, in a computer programming course for undergraduate students, teachers mostly use conventional testing to evaluate students' knowledge/skill in writing a program. Commonly, the program output determines their score. However it does not provide them with any opportunity to learn how to improve their programming experience. The students would benefit more if their coding could be analyzed. Guidance could be provided accordingly.

From a software engineer's point of view, Function Point Analysis (FPA) and Line of Code Analysis (LCA) could be used to analyze the source code. FPA, developed by Allan J. Albrecht in 1979 as a new way of quantifying software size based on the users' view of software [3]. In 1984, the International Function Point Users Group (IFPUG) was formed to promote a standardized method for measuring the function and development of the function of software applications. Based on what the user requests and receives in function return. The function point is used to count and calculate the function size by assigning weight to each individual function and the adjustment factor. This function point is proposed for checking the project growth and improvement by comparing the programming quality before and after. The amount of growth is an indication how well requirements were met and communicated to/by the project team. The number of Lines of Code has been used in "Analysis of Source Codes Created by Beginners in Programming Education"[4]as a method of evaluating program-creating ability, and the degree of difference in programmers' productivity.

In addition, the most important information and knowledge that should be delivered to another programmer or another team member are "Comments". Source code documentation is a fundamental engineering practice critical to efficient software development. It provides information of coding algorithm of the author. In other cases the source code documentation acts as a specification of behavior for other engineers. Without the comments, they are forced to get the information they need by tracing the logic of the program which is extremely difficult and error-prone.

Because of the three techniques' importance in computer programming (i.e., FPA, LCA, Comment), these three techniques could be integrated, and analysis of coding program could be diagnosed in-depth resulting in helpful guidance for undergraduate students. Consequently, this study proposed an innovative tool named CPA for computer programming which is used to

Manuscript received June 18, 2013; revised August 20, 2013.

integrate the techniques of FPA, LCA, and Comments Analysis (CA) together for analyzing source code. It may be an alternative way to evaluate coding in computer programming courses. Moreover, it could be used for monitoring students' computer programming progression. Furthermore, it could be used to analyze students' understanding of the system requirements and function requirements as well.

II. Functions for Analyzing Computer Programming

Since 1962, in computer programming area, computer scientists had attempted to find new ways to diagnose computer programming away from old fashioned debugging. Zelkovitz mentioned that the important concepts for any software engineering are system design, testing, debugging, verification, and coding program [5]. For analyzing computer programming, there are several functions such as Line of Code, Comment, and Fuction. Line of Code: SLOCCount is a set of tools for counting physical Source Lines of Code (SLOC). This tool was used in several studies to measure the SLOC. SLOCCount is a open source code under the General Public License (GPL). It has been developed by Devid A. Wheelers who used to measure SLOC of entire GNU/Linux distributions. Moreover, CLOC comes from Count Line of Code [6]. This is alternatively projected that it contains code from David Wheeler's SLOCCount. Then, the comparison of both results should be equivalent. Thus the result would be acceptable. In additional, CLOC provides function to strip out the comments. These comments have to call another function to analyze the comments to check readability. Comments should be readable written in basic English. Then, the analyzer is checking the language and sentences for human readability. This comment is the most important while working in team. If the code does not contain any comments, other programmers may not understand what and how the code works. To analyze the comments readability, Perl module using Lingua::EN::Fathom is used. This module returns score into 3 parts including Fog index, Flesch index, and Kincaid index as follows:

1. Fog index is used to analyzed text or block of sentence. It has been developed by Robert Gunning. The index indicates the number of years of formal education a reader of average intelligence would need to read the text and understand that words or sentences.
2. Flesch index is used to analyze the reading score of text file or block of sentence. This score rating is a 100 point scale. The higher the score shows easier to understanding the comment.
3. Kincaid index is used to show grade level score of text file or block of sentence. If score is 8.0, the document then can be understood by eighth grade students.

Moreover, functionis parse of code to catch the function call, for example, "include()", "mysql_connect()", "mysql_create()", etc. In the list of function, we can broadly analyze the programming code to support the system entity and protect the error, for example Db_create() needs to have Db_errorand createsa database if it failed, Db_open() needs to have Db_errorad opensa database if it failed, Db_create() needs to have Db_freeand frees a database if it created, and While_loop() needs to have exit() and exists a loop exit, etc.

We can see that these three functions are necessary for analyzing computer programming. If we have embedded analyzer which integrates these three functions to check computer programming, it could be beneficial to the monitoring of students' progression and competency in computer programming. Furthermore, it could benefit analysis of students' understanding of the system requirements and function requirements as well. Therefore, in next section, we shall propose the development of Computer Programming Analyzer (CPA) based on these three functions.

III. A Development of Computer Programming Analyzer (CPA)

In Thailand, the senior project is the last subject in which every student in software engineering is required to engage in project work in order to evaluate their knowledge prior to graduation. In this study, we provided students with system requirements and software specification including the functions. Moreover, we required students to apply PHP programming language in combination with using MySQL database to develop software. We developed CPA for analyzing their source code, provide statistic feedback and comments to them, accordingly, based on the three computer programming techniques' importance (i.e., line of code, comment, and function count). Fig. 1 shows that the analyzer will automatically analyze the source code by the 3 major components.

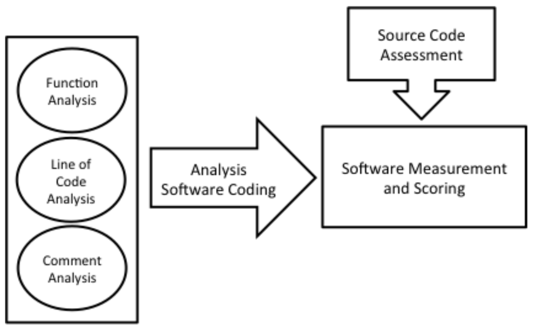


Figure 1. Illustrate the CPA System Architecture

Source code composed of pure PHP code and comments. The pure PHP code is parsed by the source code parser. This pure PHP code will be transferred to the code analyzer that is the Function Analysis and Line of Code Analysis. The Function Analysis analyzes the function process of the code. This function will be referenced using the standard declaration function in PHP. Therefore, the most of standard function in PHP code will be analyzing by Function Analysis. The project is designed to utilize a database thus database communication protocols that facilitate storage and retrieval of information. MySQL Database using PHP was chosen. Then, the Function Analysis will be referred to the standard database communication declaration. The Line of Code Analysis analyzes the pure code in terms of size counting the lines of code in the program. This analysis will be used to analyze the pure PHP code. It will show the size of code that increases with the number of features as initially defined in the program requirements. In the second part of source code analysis, the source code parser parses the comments. The number of comments in the source code will be used to make sure that the programmer took care of the algorithms' tracability. The readable index is a good quality indicator for the code comments. All the three parts of the evaluation score in terms of Analytic Scoring of the source code. The scoring indicates the programming skill. It relies on source code and the scoring of the comments. The source code assessment is the intelligence engine to identify the possible code or function that may lead to program failure. The source code assessment gives feedback to the student to either re-check or modify the code. Moreover, the clarification of the process flow of CPA is shown in Fig. 2.

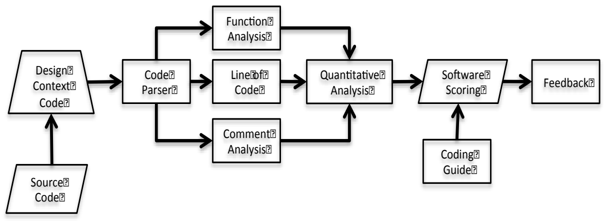
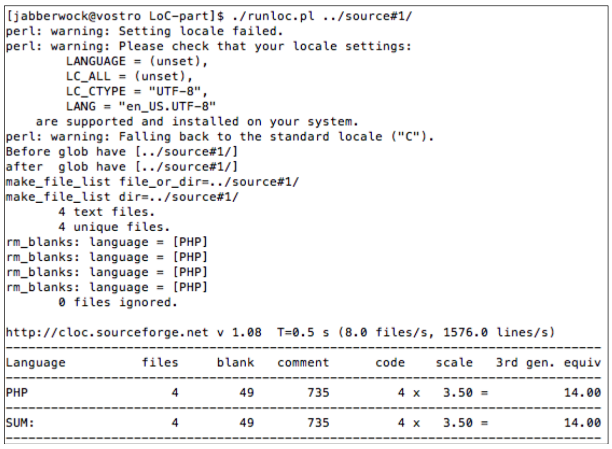


Figure 2. Illustrate the process flow of CPA


```
[jabberwock@vostro LoC-part]$ ./runloc.pl ../source#1/
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
        LANGUAGE = (unset),
        LC_ALL = (unset),
        LC_CTYPE = "UTF-8",
        LANG = "en_US.UTF-8"
    are supported and installed on your system.
perl: warning: Falling back to the standard locale ("C").
Before glob have [../source#1/]
after  glob have [../source#1/]
make_file_list file_or_dir=../source#1/
make_file_list dir=../source#1/
       4 text files.
       4 unique files.
rm_blanks: language = [PHP]
rm_blanks: language = [PHP]
rm_blanks: language = [PHP]
rm_blanks: language = [PHP]
       0 files ignored.

http://cloc.sourceforge.net v 1.08  T=0.5 s (8.0 files/s, 1576.0 lines/s)
-------------------------------------------------------------------------------
Language          files     blank   comment      code    scale   3rd gen. equiv
-------------------------------------------------------------------------------
PHP                   4        49       735         4 x   3.50 =          14.00
-------------------------------------------------------------------------------
SUM:                  4        49       735         4 x   3.50 =          14.00
-------------------------------------------------------------------------------
```
Figure 3. Command line execute code and result for Line of Code for source code

Fig. 2 shows that the source code was written by PHP code. These PHP code is one of the dynamic web page code. The web page may not be written only in one file. It could be more than 10 files. Therefore, the source codes that upload to CPA will be packed by "tar" packager. Then, the source code is analyzed by CPA. CPA has an integrated engine to identify the source code context. This is the design of the context code. This code is separately stored into the temporary storage. The PHP code uses the code parser to separate the code. The PHP code is send to Function Analysis to analyze the call of functions. Secondly, The PHP coded is analyzed in terms of Lines of Code. Lastly, the code is send to the Comment Analysis function to analyze the comments. Finally, CPA provides the scoring and statistic feedback to students and programmers as shown in Fig. 3.

IV. Results and Discussion

In this study, we recruited two undergraduate students who use PHP computer language to develop software in their senior project. After receiving the feedback from CPA, we found that, in the first version of source code, there were 4 files for completing the basic system requirements as shown in Table I. After analyzing by using the above analysis, some code was not usable and some parts of the source code might have led to system crash because of manage flow and good declaration. Following the programming suggestions provided by CPA, the second version of the source code, comprised 29 files finally complying with the system requirements. The statistics of the line of code showed an incremental around 625%. The comparison between the second and the first version implies that the increased number of files in such professional computer programming use as SummaryResult.php", "Test.php", "Testing.php", "UpdateUser.php", "UserSetting.php", "load_user.php" and "login.php" among others.

TABLE I. The Programming Improvement

	The 1st Version of Source Code	The 2nd Version of Source Code	Percentage Development Increment
Total Number of Files	4 files	29 files	625.00%
Number of Code	568	1942	241.90%
Number of Comments	171	239	39.77%
Number of blank line	49	188	283.67%

Moreover, the "Number of Code" is increased in the second version when compared to the first one from 568 to 1942 lines, an increase of 241.90%. This number clearly shows that the students were able to improve their computer programming ability. This was related to the number of files in the first program version comprising only 4 files. Then, the number of files increased to 29 after receiving feedback and comments implying that the students understood writing source code. On the other hand, the program comments explaining the computer program functions seem to be beneficial for others programmers. In the second version of source code, they found that the students have been aware of using comments during computer programming as the number of comments increased by 39.77%. In further information of blank line, it showed the significant of source code in term of coding style. Some programmers produced more lines of code by inserting blank lines. If the analyzer counted only lines of code, hence this information might loose significance in the programmers' evaluation.

In addition, Table II shows the number of function calls used in both versions of source code. In the first version, there were 217 function calls, and 737 function calls in the second version measuring an increase in function calls by 239.63%. This number is related to the percentage in progression of code development. In terms of data accessing, it showed that the students' program design in form of files direct access. But it had a little bit implements MySQL database accessing. In the second program version, the students changed the code from direct file access to MySQL database accessing. This was in line with the system requirements. It completed change the coding behavior. Using the MySQL database platform, the rule was to support the error checking and free database when it finished. In the first program version only a small number of this function calls was used. It would have been led to an execution error in case of un-detective bug. There is more development in the second program version as shown by the increase in lines of code by 683.33%. The "Exit" function would be replaced by die(). Some sort of problem was that in the debug mode did not show the exit cause. Die() is most important to relevant the unfortunately faulty. In PHP programming, there is a function that checks validity of the statement of variable declaration and returns a NULL when true. As Pavluk mentioned that "A programming language is useless without variables, PHP is no exception" [7]. This was the point to make sure that the statement of variables have been declared and used with variables returned.

TABLE II. THE NUMBER OF FUNCTION CALLED

	The 1st Version of Source Code	The 2nd Version of Source Code	Percentage Development Increment
Total Function	217	737	239.63%
Number of Files	4	29	625.00%
- Functional Available to use			
- file accessing			
- fread()	0	0	<removed>
- fwrite()	19	0	<removed>
- fopen()	6	0	<removed>
- fclose()	6	0	<removed>
- database accessing			
- mysql_connect()	3	22	633.33%
- mysql_db_query()	15	36	140.00%
- mysql_fetch_db()	15	26	73.33%
- mysql_select_db()	3	22	633.33%
- Function Reliability			
- Return Function Code			
- Database			
- mysql_error_db()	6	47	683.33%
- mysql_free_db()	0	5	500.00%
- Exit with conditioning			
- die()	6	48	700.00%
- Variable validities			
- PHP isset()	0	20	200.00%

Moreover, there are some feedbacks from the Project Manager who supported the student's senior project. It showed that increased development was related to the number of lines of code. Normally, we could check the development by running the program and analyzing the output of the system. These results implied that the students understood coding, and CPA helps examining the code with statistic number as the progress of project. Moreover, the participating students were asked to clarify their opinions of the programming suggestions from CPA. They responded in a consensus way that they were satisfied with the computer programming suggestions. Overall, the students' responses in the interviews indicated that the suggestions helped them improve their computer programming competency and helped them becoming aware not only of the causes of their computer programming barriers but also of the use of functions and the use of comments during computer programming.

V. CONCLUSION

This study developed the Computer Programming Analyzer (CPA) based on three important techniques in computer programming; line of code, comments, and the function count. CPA is used to check and monitor the student's progression and competency in computer programming. After using CPA, students showed an improvement in writing program code in PHP programming language. CPA could help not only teachers in classrooms to assess students' computer programming skills but it could also be of interest for the industry or computer engineering organizations as to analyze and improve source code in a team. The results from this study could be further used to develop an assessment programming project in the everyday classroom. It could serve progression in this field and be of ultimate evaluation for programming projects.

Moreover, the CPA can be applied to most computer programming courses or education in software engineering, such as PHP programming course, or senior project in PHP coding. When applying this approach, it is suggested that experienced teachers be invited to participate in the designof those programming projects. In developing web-based computer programming analyzer, several functions need to be taken into consideration:

(1) The Line of Code analyzer, it has been applied from CLOC and SLOCCount. The CLOC, it comes from Count Line of Code by Al Danial[6]. It counts blank lines, comment lines, and physical line of source code in most common programming language. CLOC has contains code from SLOCCount by DevidWheller. CLOC will be used as core analyzer of source code.

(2) The Function Analyzer, it has been derived from Perl programming. Regular Expression will be used to parse the Function code. It will used to print out the detected function. This print out will be used to analyze the function call of source code.

(3) The Comment Analyzer, it is the integrated function from Lingua::EN::Fathom module from Perl's CPAN. It requires function from CLOC to strip out the comments. Then it will analyze these comments in terms of read abilities index. This read abilities will show in form of Fog Index, Flesch reading ease score and Flesch-Kincaid grade level score.

(4) The Parse::RecDescent developed by Damian Conway fromperl's CPAN is need. This parser can identify the grammar for PHP code. Therefore, it

has limitation to parse the source code only PHP code by standard definition of PHP.

(5) The analyzer will be running on Linux command line with installed Perl Interpreter. CentOS version 6 is the development operating system. Perl version 5 will be added in CentOS as development tools.

(6) With the result of Lines of Code being analyzed the coding trend and the growth of coding are shown. This indicator will be used to investigate the development time and complexity of code.

(7) With the result Function Analyzed, it shows the count number of function. Any related code, it need to identify such as Database accessing, File Accessing, or Function close.

(8) With the Comment Analyzed, it shows the readable of comments. This indicator shows as coding documents. It can be show the readability of comment in source code.

In additional, to introduce the CPA into the other practitioners/researchers, a step-by-step lead-in procedure as given following:

Step 1: Give user requirements to student to design and develop the program by face-to-face.

Step 2: Student submits the source code.

Step 3: Teacher control the analyzer to analyze the submitted code. Teacher can see the progress in development by the Line of Code analyzer. With the Function analyzer, the teacher can check the function calling. And the Comment Analyzer helps the teacher check the readabilities of comments.

Step 4: The student receives the guidance from the analyzer and conduct the student's programming competency based on the guidance. Students will be made aware of their shortcomings in programming be the analyzer. Then, the student is able to improve based on mentioned points.

Step 5: After finish the program coding, student can resubmit the source code. The teachers will see the project development after analyzing the source code.

Reference

[1] S. Drew, L. Thorpe, and, P. Bannister, "Key skills computerised assessments. guiding principles," *Assessment & Evaluation in Higher Education*, vol. 27, no. 2, pp.175-186, 2002.

[2] M. S. W. Lam, E. Y. K. Chan, V. C. S. Lee, and Y. T. Yu, "Designing an automatic debugging assistant for improving the learning of computer programming hybrid learning and education," in *Proc. 1st International Conference on Hybrid Learning and Education*, Hong Kong, PRC, 2008, pp. 359-370.

[3] A. J. Albrecht, "Measuring application development productivity." in *Proc. of the IBM Applications Development Joint SHARE/GUIDE/IBM Symposium*, CA, USA, 1979, pp. 83-92.

[4] T. Nakashima, C. Matsuyama, and N. Ishii, "Analysis of source codes created by beginners in programming education," presented at the Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007), Eighth ACIS International Conference, Qingdao, China, 2007.

[5] M. V. Zelkowitz, "What have we learned about software engineering?" *Commun. ACM*, vol. 55, no. 2, pp. 38-39, 2012.

[6] A. Danial, *Count Lines of Code (CLOC)*, Northrop Grumman Corporation: USA. 2006.

[7] M. Pavlak, "PHP Basics on the System," *System iNEWS*, vol. 340, pp. 1-4, 2008.

Suchot Sinthsirimana is a MSc. Candidate in Science and Technology Education program at the Institute for Innovative Learning, Mahidol University, Thailand.

PatcharinPanjabureeis a lecturer at Institute for Innovative Learning, Mahidol University, Thailand. She is interested in computer science education, mathematics education, computer-assisted testing, expert systems, and knowledge engineering. She is a corresponding author in this paper.