A Logical Error Detector for Novice PHP Programmers

Tung Nguyen and Caslon Chua
Department of Computer Science and Software Engineering
Swinburne University of Technology
Melbourne, Australia
{thanhnguyen, cchua}@swin.edu.au

***Abstract*— Currently PHP is the most widely used web programming language for websites [1]; however there seems to be a limited availability of debugging tools that a novice programmer can use. In this work, we propose a framework to identify logic errors committed by novice PHP programmers and prototype application to automate the process of detecting such errors. The aim is to develop a tool that may be used to assist novice programmers in their learning process, and contribute to computer science education research.**

Keywords— debugging; novice programming

I. INTRODUCTION

There have been many attempts in helping programmers debug and reduce errors in various programming languages. Weragama et. al. [2] developed a knowledge base for an Intelligent Tutoring System (ITS) for beginning students studying PHP. Kummerfeld et. al. [3] investigated how beginner programmers detect syntax errors and how to help them fix those errors. They also made a web based reference guide which is a catalogue of common C/C++ syntax errors.

Most researchers focus on programming languages that are strictly typed such as Java and C where variable must be declared before it is used and it can only be one type throughout the its scope. This allows the detect errors relating to variables during compilation stage. In contrast, PHP allows a variable to be declared and used anywhere in the program. Variable can also change to any type after it is assigned a value. In addition, a script that contains PHP codes are often mixed with HTML codes. Novice programmers can easily become confused with the structure of the program. Furthermore, debugging tools for PHP are also limited. Thus it is quite difficult for novice programmers to detect errors in PHP. We propose a framework to identify logical errors in PHP programs, and develop an application was developed as a proof of concept.

II. IDENTIFYING LOGICAL ERRORS IN PHP PROGRAMS

In the process of identifying logical errors in PHP programs, we initially focus on three types of expression formulation, namely equality condition formulation, while-loop condition formulation and for-loop expression formulation. Table I summarises the cases of the logical errors that can be detected.

TABLE I. CASES OF LOGICAL ERRORS DECTECTED

Case	Code Excerpt	Logical Error Detected

1	$number = 10;
	if ($number = 1)
	{
	echo "Value of number is 1";
	}
	else
	{
	echo "Value of number is 10";
	}
		In the condition, assignment operator (=) is used instead of comparison operator so the program will not output a correct result.

2	$i = 0;
	while ($i = 10)
	{
	$i++;
	}
		The loop will execute only once because an assignment operator is wrongly used in the condition.

3	$i = 0;
	while ($ i < 10);
	{
	$i++;
	}
		The loop will execute infinitely because the semicolon was added to the while loop by mistake.

4	$number = 0;
	for ($i = 0; $i < 10; $i++);
	{
	$number++;
	}
	echo $number;
		The loop will run infinitely because the semicolon was added to the for-loop by mistake.

5	for ($i = 0; $i < 10; $i–)
	{
	echo "$i ";
	}
		The loop will run infinitely because the iterator will be always less than 10.

6	for ($i = 20; $i < 10; $i++)
	{
	echo "$i ";
	}
		The loop will not execute any time because the iterator is greater than 10.

A. Equality Condition Formulation

For equality condition formulation errors, we focused on the use of equality operator (==) where novice programmer may at times uses the assignment operator (=). Another problem is on the use strict equality (===) over equality (==), where novice programmer may not realise the implication of functions returning numeric result if the operation successful, and a value false if otherwise. The issue is that a numeric value of 0 is treated as false in a Boolean expression.

B. While-loop Condition Formulation

For while-loop construction, we currently only checked how the condition is formulated for the while and do-while loops.

C. *For Loop Expression Formulation*

In for-loop construction, we only looked at the expression formulated in a for-loop, more specifically on assigning the starting value, loop termination condition and changing the iterator value.

III. Framework, Application, and Testing

The framework shown in Figure 1 employs a rule-based technique that accepts a PHP program input and searches the database for logical error characteristics. In addition, it also has a PHP interface that will execute the submitted program to obtain the final values of variables used.

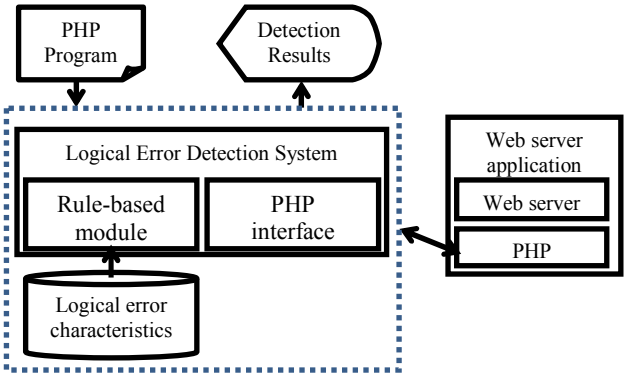

Fig. 1. Framework

The application employs a web interface (Figure 2) that allows users to submit a PHP file for checking with an option to display the final values of variables. Logical error detection starts when the user clicks on the submit button.

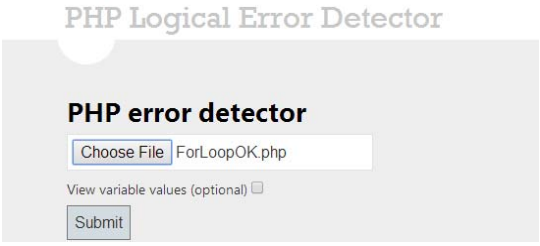

Fig. 2. Application Interface

The application will identify possible logical errors and display an error message summary to the user (Figure 3). It will also display the final variable values if users selected that option.

To assist the novice programmer in debugging their program, a tutorial-type explanation (Figure 4), which discusses the error in detail and how it can be fixed, is provided through a view link.

In testing the application, twelve small PHP programs were written to contain various errors similar to cases mentioned in Table I. Each program may contain none or several errors. In all test cases, the application was able to correctly detect the errors in PHP programs and display the expected final values of all variables.

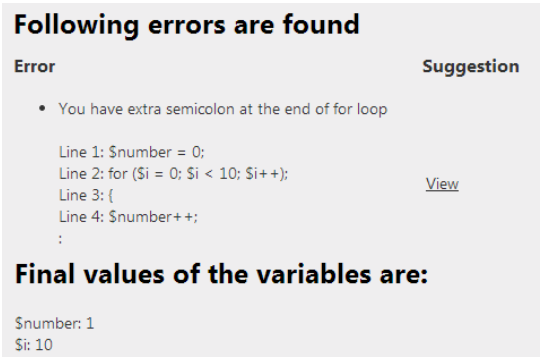

Fig. 3. Error Message Summary

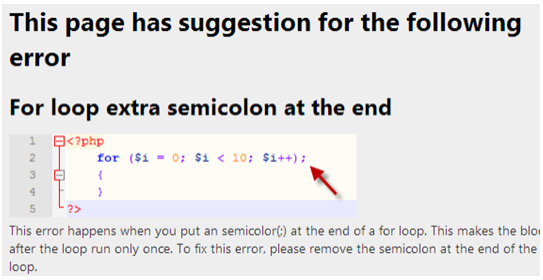

Fig. 4. Tutorial type explanation

IV. Conculsion and Future Work

In this preliminary work, a framework to identify logical errors in PHP programs was developed and an application was successfully implemented and tested.

The next phase is to conduct a data collection study to identify the various types of errors a novice programmer commits. Results of this phase will be used to enhance the logical error characteristics database, in order to increase the number of errors that the application can detect.

Finally, a user study will be conducted on usefulness of the detection system in assisting novice PHP programmers in their learning, more specifically students pursing the web programming unit for the first time. Results from phase will contribute to education research specifically on teaching programming.

References

[1] Q.-S. W. based Services, "Usage of server-side programming languages for websites," February 2014. [Online]. Available: http://w3techs.com/technologies/overview/programming_language/all

[2] D. Weragama and J. Reye, "Designing the knowledge base for a php tutor," in Proceedings of the 11th International Conference on Intelligent Tutoring Systems, ser. ITS'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 628–629.

[3] S. K. Kummerfeld and J. Kay, "The neglected battle fields of syntax errors," in Proceedings of the Fifth Australasian Conference on Computing Education - Volume 20, ser. ACE '03. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2003, pp. 105–111.[Online]. Available: http://dl.acm.org/citation.cfm?id=858403.858416.