# Review the Tests Performed on Web Applications Written with PHP - Survey

Omid Rashnoodi
Dept. of Computer, Persian Gulf International Branch, Islamic Azad University, Khoramshahr, Iran

Reza Ebrahimpour Razaz
Graduate Student of the Nonprofit Institute of Arvandan, Khorramshhr, Khozestan, Iran

Khaled Mohammadnejad
Dept. of Computer Engineering, Nonprofit Institute of Arvandan, Khorramshhr, Khozestan, Iran

## ABSTRACT

Today's use of Web applications has become a challenge because users expect to use a program that does not expose their information and has a good security and also has the speed, proper operation and high accountability. Therefore, it is imperative to use a powerful program for designing the Web and performing Web tests to verify the validity of Web applications.

Software testing has been widely used in the industry as a quality assurance technique for the various artifacts in a software project, including the specification, the design, and source code [1].The purpose of this study is to review the PHP tests in the last two decades and also evaluating the results of the tests performed on the programs written with the PHP programming language.

## Keywords

Software testing, PHP, Web applications, Web testing, performance testing, Survey

## 1. INTRODUCTION

Today's use of Web applications has become a challenge because users expect to use a program that does not expose their information and has a good security and also has the speed, proper operation and high accountability. Satisfaction of users is an important and has high importance in developing a Web application; therefore, the use of powerful programming language, such as PHP, for designing a Web application, as well as the regular execution of necessary tests to enhance the capabilities of web applications to achieve this, seems necessary. A web application should have the proper functionality in different modes. In fact, it always has to offer the best answer.

In software testing, a suite of test cases is designed to test the overall functionality of the software whether it conforms to the specification document or exposes faults in the software (e.g., functionality or security faults). However, contrary to the preconceived notion that software testing is used to demonstrate the absence of errors, testing is usually the process of finding as many errors as possible and thus improving assurance of the reliability and the quality of the software [1]. Web applications are used by virtually all organizations in all sectors, including education, health care, consumer business, banking and manufacturing, among others. Thus, it is important to ensure that the Web applications developed are properly tested due to the importance and the sensitivity of the information stored in databases of such Web applications [2,3]. PHP is a powerful programming language for creating dynamic and dynamic Web sites, a server-side language whose scripts run on the server. By PHP, you can powerfully designing and programming Web applications [4].

## 2. TEST CONCOLIC FOR PHP

Concolic testing, a technique that combines symbolic and concrete random execution to improve testing effectiveness [4]. The purpose of the Concolic Test is to participate through the combination of implicit and symbolic performances. In this type of test Random entries are imported into web applications to discover additional and alternative paths in the Web application due to different inputs [5-6-7]. Concolic (concrete and symbolic) testing techniques automate test input generation by combining the concrete and symbolic (concolic) execution of the software under test. Most test input generation techniques use either concrete execution or symbolic execution that builds constraints and is followed by a generation of concrete test inputs from these constraints [9]. Concolic testing, on the other hand, combines both these techniques, which take place simultaneously. The goal in concolic testing is to generate different input data which would ensure that all paths of a sequential program of a given length are covered. The program graphs, which depict program statements and the program execution, are provided as inputs [9].

## 3. CONCRETE, SYMBOLIC EXECUTION

Concolic testing uses concrete values as well as symbolic values for input and executes a program both concretely and symbolically, called concolic execution. The concrete part of concolic execution is where the program is normally executed with concrete inputs, drawn from random testing. The symbolic part of concolic execution collects symbolic constraints over the symbolic input values at each branch point encountered along the concrete execution path. At the end of the concolic execution, the algorithm com- puts a sequence of symbolic constraints corresponding to each branch point. The conjunction of these symbolic constraints is known as path constraints. More formally, a path constraint (PC) is defined as the conjunction of conditions on variables as a result of executing the Web application with concrete values. All input values that satisfy a given path constraint will cover the same execution path. Concolic testing first generates random values for primitive inputs and the NULL value for pointer inputs. The algorithm then executes the program concolically in a loop. At the end of execution, a symbolic constraint is negated in the original path constraint (which contains a conjunction of symbolic constraints) and the alternative branches of the program are explored. The algorithm is continued with the newly generated concrete inputs for the new path constraint. As a result, concolic testing combines random testing and symbolic execution, thus overcomes the limitations of random testing, such as the inefficient and ad hoc nature of the test cases generated [10], the difficulty in traversing the different paths in a program, redundant test input data which lead to the same observable program behaviors [12], and the low coverage obtained (due to the random nature of the input data) [11].

## 4. A STRING-BASED CONCOLIC TESTING APPROACH FOR PHP APPLICATION

Wassermann et al. develop a concolic testing-based framework for detecting SQL injection defects through dynamic test input generation by manipulating string operations in PHP. String operations are modelled using finite state transducers. Constraints are then used to generate new string values. An algorithm is then used to validate whether a string constitutes an SQL injection attack. Finally, backward slices are dynamically constructed at runtime to enable testing beyond unit level. Scripting languages such as PHP support met programming capabilities and hence are more dynamic. For instance, PHP allows function call names and variable names to be constructed dynamically, from user inputs. The presence of such dynamic language features makes it very hard for static analysis tools to analyses PHP programs. The approach tackles this challenge by using a concolic approach that records variable values in concrete execution and use them as constraints to generate new inputs symbolically [8].

In this framework, new string-typed test inputs are generated from current string values through the use of constraints. The generation of such constraints is enabled by modelling string operations and type casts as finite state transducers (FSTs), finite state machines with an input tape and an output tape. As in concolic testing, constraints are generated, solved and inverted to generate new test inputs to cover different paths of the PHP program under test [8].

The state space of any non-trivial program may be too large for a concolic testing technique to handle efficiently. Sometimes, significant portions of a program's execution, such as logging, are not relevant to the properties of interest. This problem is alleviated by analyzing program points that are (directly or indirectly) relevant to possible failure, in a backward manner. Starting at function calls that send a query to the database, other functions where this call occurs are iteratively added. Control dependency and data dependency are resolved by maintaining a stack trace of function calls, and by examining symbolic values during execution. Approximately this process constructs a backward program slice, and is shown to dramatically reduce the number of constraints generated, sometimes by several orders of magnitude [8].

## 5. APOLLO: A PATH-BASED CONCOLIC TESTING FRAMEWORK FOR PHP THEN THE ADDITIONAL PATH CONSTRAINTS ARE ADDITIONAL

Artzi offers a concolic test technique for the PHP web application. The path restrictions generated by symbolic execution are stored in a queue; the queue begins with an empty path restriction. A restrictor is used to find the implicit input that meets the path limit taken from the queue. The program runs implicitly on the input, and tested for failure and error. The path and input limits for each detected failure are merged.in the report. The program is now implemented symbolically with the same implicit input values ( That are selected by the constraint eliminator ) And limitations of different paths is obtained (That is, a Boolean expression with the conjunction of the conditions that are correct, when the application runs with the input) New test inputs are created by solving modified versions of the resulting path restrictions, If there is a solution to the alternative path restriction that matches the input, the execution of the program is completed with the opposite branch[7].

## 6. EVALUATION OF PERFORMANCE TESTING IN APPLICATIONS WRITTEN WITH PHP

To evaluate PHP's performance, a scenario was developed that included a simple logical page using a three-layer architecture, In order to obtain the pass rate, response time, percentage of CPU utilization and percentage of unemployment time, the performance test was performed [13]. The performance test is used to evaluate a program that is able to run under expected and peak conditions and for a capacity increase scale. Performance test, the process of measuring the performance of a program depends on a given set of conditions and inputs. In order to test performance, the program should be expanded and deployed in infrastructures similar to the production environment [14]. To run the test, WAPT Pro software was used by Soft Login Company, This software will test virtual users instead of real client, and In other words, it implements applications by downloading tests by virtual users. And confirms whether it supports the tested traffic levels, patterns, and combinations or not. This method is typically used to evaluate the software's ability to perform under expected loads, determine the system capacity and identify the source of each tag [15]. Three performance tests, such as load tests, stress testing and capacity testing, were carried out in this scenario[14].

Load Test: Usually this test is used to verify the behavior of applications under normal and peak times and in order to investigate the web application, it can provide the desired performance goals these goals are often outlined in the service level agreement or the collection of document requirements. The load test is able to measure response times, power levels, utilization levels of resources (memory, CPU, disk input and output, and network input / output) [14]. Stress Test: This test is used to assess the behavior of programs at times that are beyond the normal condition [14]. Capacity Test: This is a test for load testing and determining the final failure point of the host server [14]. The goal was to measure the performance of a Web application written in PHP programming language Implemented with WAPT pro software and a virtual user sample of 1, 25, 50, 75, 100 virtual users were created [16]. The process of carrying out the load test was performed according to Microsoft's standard [17].

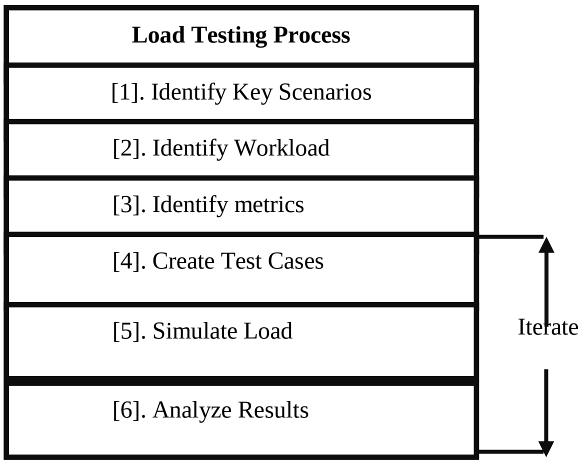

**Fig. 1: Steps to Perform a Load Test in accordance with Microsoft® Standard [17]**

In order to evaluate the programming language of PHP. The results were compared to other powerful programming language software called asp.net. The same scenario with the same quality, implemented on similar systems and implemented with asp.net [13]. The results showed that, in terms of response time and memory consumption, implementation using PHP is better than asp.net. Of course, this assessment was conducted solely to measure ineffective goals[16].

**Table 1: Comparison of response time in using PHP and asp.net [16]**

| Users | Response time(my) | |
|---|---|---|
| | PHP | asp.net |
| 1 | 1.98 | 3.0682 |
| 25 | 4.621 | 4.865 |
| 50 | 15.482 | 57.452 |
| 75 | 20.952 | 65.652 |
| 100 | 34.768 | 76.9 |

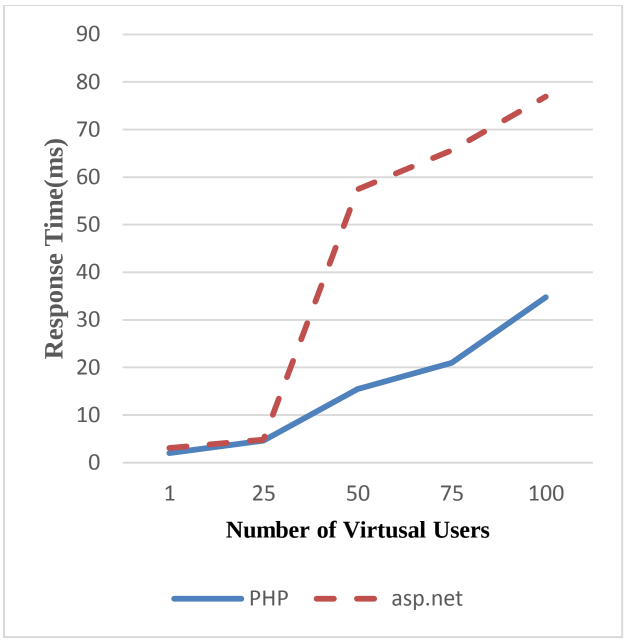

**Fig. 2: Chart Response time in using PHP and asp.net [16]**

**Table 2: Comparing Memory Productivity Using PHP and Asp.net [16]**

| Users | Memory Utilization(MB) | |
|---|---|---|
| | PHP | asp.net |
| 1 | 755 | 1032 |
| 25 | 796 | 1134 |
| 50 | 864 | 1282 |
| 75 | 985 | 1325 |
| 100 | 1185 | 1376 |

.

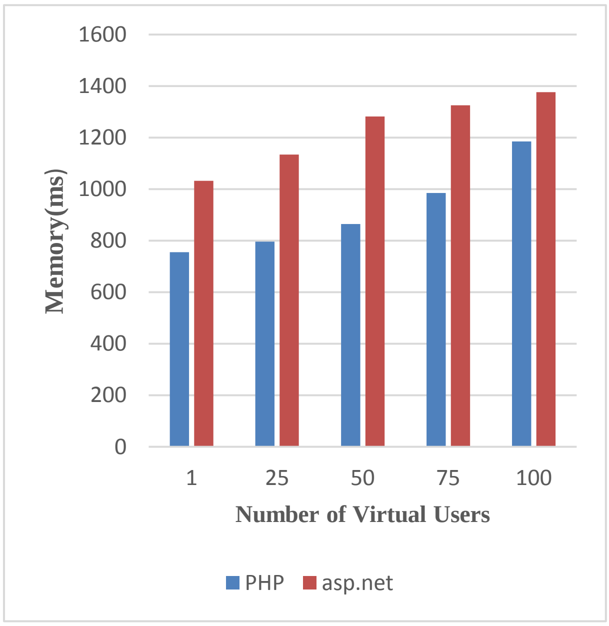

**Fig. 3: Chart of memory usage in using PHP and asp.net [16]**

## 7. CONCLUSIONS

In this research, the testing methods performed on web applications written with the PHP programming language were reviewed and the goal was to review the tests that took place in the last few years on the powerful PHP programming language and in an experiment this programming language was compared to another programming language called asp.net. Finally, this review and comparison showed that PHP has better response time and memory usage than asp.net, and it seems more appropriate to meet non-operational goals.

## 8. REFERENCES

[1] G.J. Myers, C. Sandler, T. Badgett, 2011. The Art of Software Testing, Wiley, New Jersey, USA.

[2] S. Kals, E. Kirda, C. Kruegel, N. Jovanovic, SecuBat, 2006. a web vulnerability scanner, in: Proceedings of the 15th International Conference on World Wide Web, WWW'06, ACM, New York, NY, USA, pp. 247–256.

[3] S. Sampath, V. Mihaylov, A. Souter, L. Pollock, 2004. A scalable approach to user-session based testing of Web applications through concept analysis, in: Proceedings of the 19th International Conference on Automated Software Engineering, pp. 132–141.

[4] R.J. Lerdorf, K. Tatroe, B. Kaehms, R. McGredy, Programming PHP, 1st ed. O'Reilly & Associates, 2002. Inc, Sebastopol, CA, USA.

[5] P. Saxena, D. Akhawe, S. Hanna, F. Mao, S. McCamant, D. Song, 2010. A symbolic execution framework for JavaScript, in: Proceedings of the 2010 IEEE Symposium on Security and Privacy, SP'10, IEEE Computer Society, Washington, DC, USA, pp. 513–528.

[6] L. Zhou, J. Ping, H. Xiao, Z. Wang, G. Pu, Z. Ding, 2010. Automatically testing web services choreography with assertions, in: Proceedings of the 12th International Conference on Formal Engineering Methods and Software Engineering, ICFEM'10, Springer-Verlag, Berlin, Heidelberg, pp. 138–154.

[7] S. Artzi, A. Kiezun, J. Dolby, F. Tip, D. Dig, A. Paradkar, M.D. Ernst, 2008. Finding bugs in dynamic web applications, in: Proceedings of the 2008 International Symposium on Software Testing and Analysis, ISSTA'08, ACM, New York, NY, USA, pp. 261–272.

[8] G. Wassermann, D. Yu, A. Chander, D. Dhurjati, H. Imamura, Z. Su, 2008. Dynamic test input generation for web applications, in: Proceedings of the 2008 International Symposium on Software Testing and Analysis, ISSTA'08, ACM, New York, NY, USA, pp. 249–260.

[9] K. Sen, 2007. Concolic testing, in: Proceedings of the Twenty-second IEEE/ ACM International Conference on Automated Software Engineering, ASE'07, ACM, New York, NY, USA, pp. 571–572.

[10] B.P. Miller, G. Cooksey, F. Moore, 2006. An empirical study of the robustness of Mac OS applications using random testing, in: Proceedings of the 1st International Workshop on Random Testing, RT'06, ACM, New York, NY, USA, pp. 46–54.

[11] P. Godefroid, N. Klarlund, K. Sen, 2005, DART: directed automated random testing, SIGPLAN Not. 40, p. 213–223.

[12] K. Sen, D. Marinov, G. Agha, CUTE, 2005. A concolic unit testing engine for C, in: Proceedings of the 10th European Software Engineering Conference held jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering, ESEC/FSE-13, ACM, New York, NY, USA, pp. 263–272.

[13] Ch Ram Mohan Reddy, D Evangelin Geetha, KG Srinivasa, 2011. Eearly performance prediction of web services, International Journal on Web Service Computing (IJWSC), Vol.2, No.3.

[14] Osama Hamed, Nedal Kafri, 2009. "Performance Prediction of Web Based Application Architectures Case Study: .NET vs. Java EE", International Journal of Web Applications, Vol.1.

[15] WAPT pro, Available at: http://www.softlogica.com.

[16] Meier, J. D., Vasireddy, S., Babbar, A., Mackman, A, 2004. Improving .NET Application Performance and Scalability, Patterns & Practices". Microsoft Corporation, ISBN 0-7356-1851-8.

[17] Microsoft web application stress tool, MSDN library, Available at: http://www.microsoft.com.