

## **WEB SCRAPING BASED TEST CASE GENERATION FOR LOAD AND STRESS ANALYSIS IN THE SERVERS**

*Parmeet Mann*

*Research Scholar*

*Shri Venkateshwara University*

*Gajraula, Amroha, Uttar Pradesh, India*

*Prof. A. K. Vasishtha*

*Shri Venkateshwara University*

*Gajraula, Amroha, Uttar Pradesh, India*

### **ABSTRACT**

Test Case Generation is one of the indispensable task in programming planning that is having a course of action of conditions under which a tester will make sense of if an application, programming system or one of its components is filling in as it was at first settled for it to do. The segment for making sense of if an item extend or structure has completed or failed such a test is known as a test prophet. In a couple of settings, a prophet could be an essential or use case, while in others it could be a heuristic. It might take various test cases to confirm that an item extend or structure is considered sufficiently analyzed to be released. Test cases are regularly insinuated as test scripts, particularly when made - when they are by and large accumulated into test suites. Monte Carlo schedules (or Monte Carlo investigations) are a far

reaching class of computational figurings that rely on upon repeated unpredictable testing to get numerical results. They are routinely used as a piece of physical and experimental issues and are most useful when it is troublesome or hard to use other numerical procedures. Monte Carlo procedures are mainly used as a piece of three specific issue classes: change, numerical compromise, and making draws from a probability scattering. In the proposed work, we will make a progression of test cases to evaluate the execution and uprightness of web application so that the synchronous and high load applications can be executed with no overhead.

Keywords - Software Testing, Test Case Generation, Monte Carlo Based Test Case Generation

## **INTRODUCTION**

Software testing technique incorporates the execution of a product section or structure fragment to evaluate one or more properties of side interest. When in doubt, these properties demonstrate the extent to which the part or system under test:

- Meets the essentials that guided its framework and progression,
- Responds viably to an extensive variety of inputs,
- Performs its abilities within an agreeable time,
- Sufficiently usable,
- installed and continue running in its proposed surroundings, and
- Achieves the general result its accomplices wish.

To propose and protect the examination work, various exploration papers are dissected. Taking after are the passages from the distinctive examination work performed by number of academicians and analysts.

- [1] With the developing multifaceted nature of web applications, distinguishing web interfaces that can be utilized for testing such applications has turned out to be progressively testing. Numerous procedures that work viably when connected to basic

- web applications are deficient when utilized on advanced, dynamic web applications, and might at last result in lacking testing of the applications' usefulness. To address this issue, we display a system for naturally finding web application interfaces taking into account a novel static investigation calculation. This work likewise report the consequences of an experimental assessment in which we analyze our method against a conventional methodology. The aftereffects of the correlation demonstrate that our strategy can (1) find a higher number of interfaces and (2) produce test inputs that accomplish higher scope.
- [2] In this paper, the creators create routines that utilization logged client information to construct models of a web application. Logged client information catches dynamic conduct of an application that can be helpful for tending to the testing issues of testing web applications. This methodology consequently assembles measurable models of client sessions and naturally gets test cases from these models. This work gives a few option displaying approaches taking into account measurable machine learning strategies. We explore the adequacy of the test suites produced from our techniques by performing a preparatory study that assesses the created test cases. The aftereffects of this study exhibit that our methods can create test cases that accomplish high scope and precisely show client conduct. This study gives bits of knowledge into enhancing our systems and persuades a bigger study with a more different arrangement of utilizations and testing measurements.
  - [4] Modified condition/choice scope is a basic scope foundation requiring that every condition inside of a choice is appeared by execution to autonomously and effectively influence the result of the choice. This foundation was produced to address the issue for broad testing of complex Boolean expressions in wellbeing basic applications. The paper portrays the altered condition/choice scope basis, its properties and ranges for further work.

- [5] Pairwise testing is a detail based testing measure, which requires that for every pair of data parameters of a framework, each mix of substantial estimations of these two parameters be secured by no less than one experiment. In this paper, the work propose another test era procedure for pairwise testing.
- [6] Test case prioritization strategies plan test cases for execution in a request that endeavors to amplify some goal capacity. An assortment of target capacities are pertinent; one such capacity includes rate of deficiency identification a measure of how rapidly blames are recognized inside of the testing process. An enhanced rate of issue location amid relapse testing can give quicker criticism on a framework under relapse test and let debuggers start their work sooner than might somehow or another be conceivable. In this paper the work depict a few strategies for organizing test cases and report our exact results measuring the adequacy of these procedures for enhancing rate of shortcoming discovery. The outcomes give bits of knowledge into the tradeoffs among different methods for experiment prioritization
- [7] Pairwise testing has turned into a crucial apparatus in a software analyzer's toolbox. This article gives careful consideration to convenience of the pairwise-testing system. Most devices, notwithstanding, need useful elements that are vital for them to be used in the industry. This article gives careful consideration to convenience of the pairwise-testing method. In particular, it doesn't depict any profoundly new technique for productive era of pairwise test suites—a subject that has as of now been explored widely—neither does it allude to any particular contextual investigations or results that have been gotten through this strategy of test-case era. It focuses on routes in which the immaculate pairwise-testing approach

## **ANALYSIS OF ASSORTED TESTING PARADIGMS**

International Refereed Journal of Reviews and Research

Volume 3 Issue 1 January-February 2015

International Manuscript ID : 23482001V3I101012015-10

(Approved and Registered with Govt. of India)

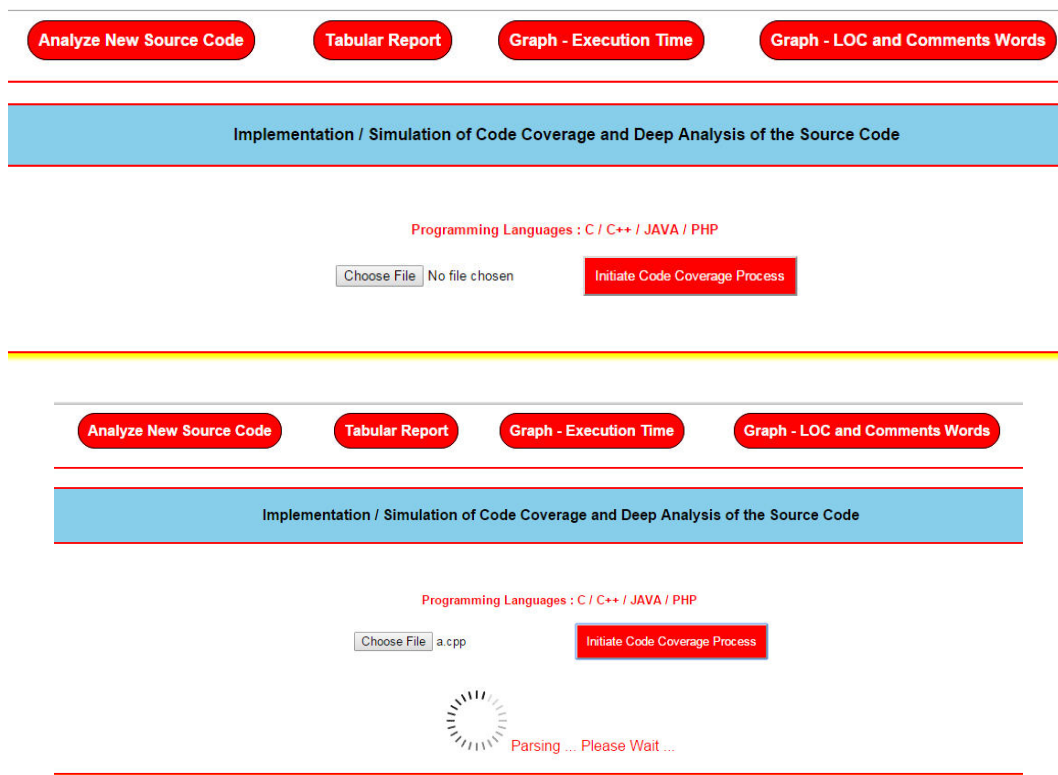
Test Type / Test Environment	Development Operating Environment	Target Operating Environment	Production Operating Environment
Software Interface Testing	●	●	●
HW/SW Integration Testing	●	●	●
Functional Requirements Verification	●	●	●
Multi-Component Thread Testing	●	●	
Performance/Fail-Over Testing	●	●	●
Stress Testing		●	●
Security Testing		●	●
System Certification		●	●
Test Data Analysis	●	●	●

Type	Scope	Tools / Methods
Unit Testing	class, method	tool: Junit
Component Testing	Component, including interface among different classes	methods: stub, driver
Integration Testing	Different Components and interface among them	methods: stub, driver
System Testing*	The system as a whole	tool: programmers' pride (Just kidding)
Acceptance Testing	Customers' feedback or experience of using your software	methods: survey etc.

\* The difference between system testing and integration is very simple to understand but often ignored. But you should understand the difference because it is really very simple. I will manage to explain it in the following 3 lines: if you have the software there already, you play around with it, so you are doing system testing. Otherwise if you are still on the stage of testing whether stand-alone components can work as a whole, you are doing integration testing.

For implementation and simulation, a web based simulator is developed. Using the simulator, any number of test cases can be generated and tested on the HTTP Web Server.

International Refereed Journal of Reviews and Research  
Volume 3 Issue 1 January-February 2015  
International Manuscript ID : 23482001V3I101012015-10  
(Approved and Registered with Govt. of India)



**Figure 1 – Implementation Screenshot**

### Parsing Results and Performance Analysis

Total Lines of Code (Including Blank Lines) : 9

Total Comments : 1

Total Words in the Comments : 5

Total Words : 20

Percentage (Density) of Comment Lines :  $5/20 \times 100 = 25\%$

Total Executable Lines (Without Comments and Blank Lines) : 3

*Stop Words in the Repository : NULL, VARIABLE, INTEGER, RESULTS, RETURN, cool, hot, watch, look, hard, soft, awesome, amazing, super, while, do, for, if, else, return, switch, suck, i, me, we, our, buddy, dude, superb, !, @, \*, #, \$, %, ^, &, (, ), +, -, ^, `, ~, <, >, ?, {, }, [, ], 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, bugging, mine, abstract, assert, boolean, break, byte, case, catch, char, const, continue, default, double, enum, extends, final, hey, finally, flot, goto, implements, import, instanceof, int, interface, long, native, new, package, private, protected, public, return, short, static, staticfp, super, switch, synchronized, this, throw, throws, transient, try, void, volatile, while, false, null, true, inheritance, debugging, error, true, false, auto, break, case, char, const, continue, default, do, double, else, enum, extern, float, for, goto, if, int, long, register, return, short, signed, sizeof, static, struct, switch, typedef, union, unsigned, void, volatile, while, about, above, across, after, afterwards, again, against, all, almost, alone, along, already, also, although, always, am, among, amongst, amoungst, amount, an, and, another, any, anyhow, anyone, anything, anyway, anywhere, are, around, as, at, back, be, became, because, become, becomes, becoming, been, before, beforehand, behind, being, below, beside, besides, between, beyond, bill, both, bottom, but, by, call, can, cannot, cant, co, computer, con, could, couldnt, cry, de, describe, detail, do, done, down, due, during, each, eg, eight, either, eleven, else, elsewhere, empty, enough, etc, even, ever, every, everyone, everything, everywhere, except, few, fifteen, fifty, fill, find, fire, first, five, for, former, formerly, forty, found, four, from, front, full, further, get, give, go, had, has, hasnt, have, he, hence, her, here, hereafter, hereby, herein, hereupon, hers, herself, him, himself, his, how, however, hundred, i, ie, if, in, inc, indeed, interest, into, is, it, its, itself, keep, last, latter, latterly, least, less, ltd, made, many, may, me, meanwhile, might, mill, mine, more, moreover, most, mostly, move, much, must, my, myself, name, namely, neither, never, nevertheless, next, nine, no, nobody, none, noone, nor, not, nothing, now, nowhere, of, off, often, on, once, one, only, onto, or, other, others, otherwise, our, ours, ourselves, out, over, own, part, per, perhaps, please, put, rather, re, same, see, seem, seemed, seeming, seems, serious, several, she, should, show, side, since, sincere, six, sixty, so, some, somehow, someone,*



International Refereed Journal of Reviews and Research

Volume 3 Issue 1 January-February 2015

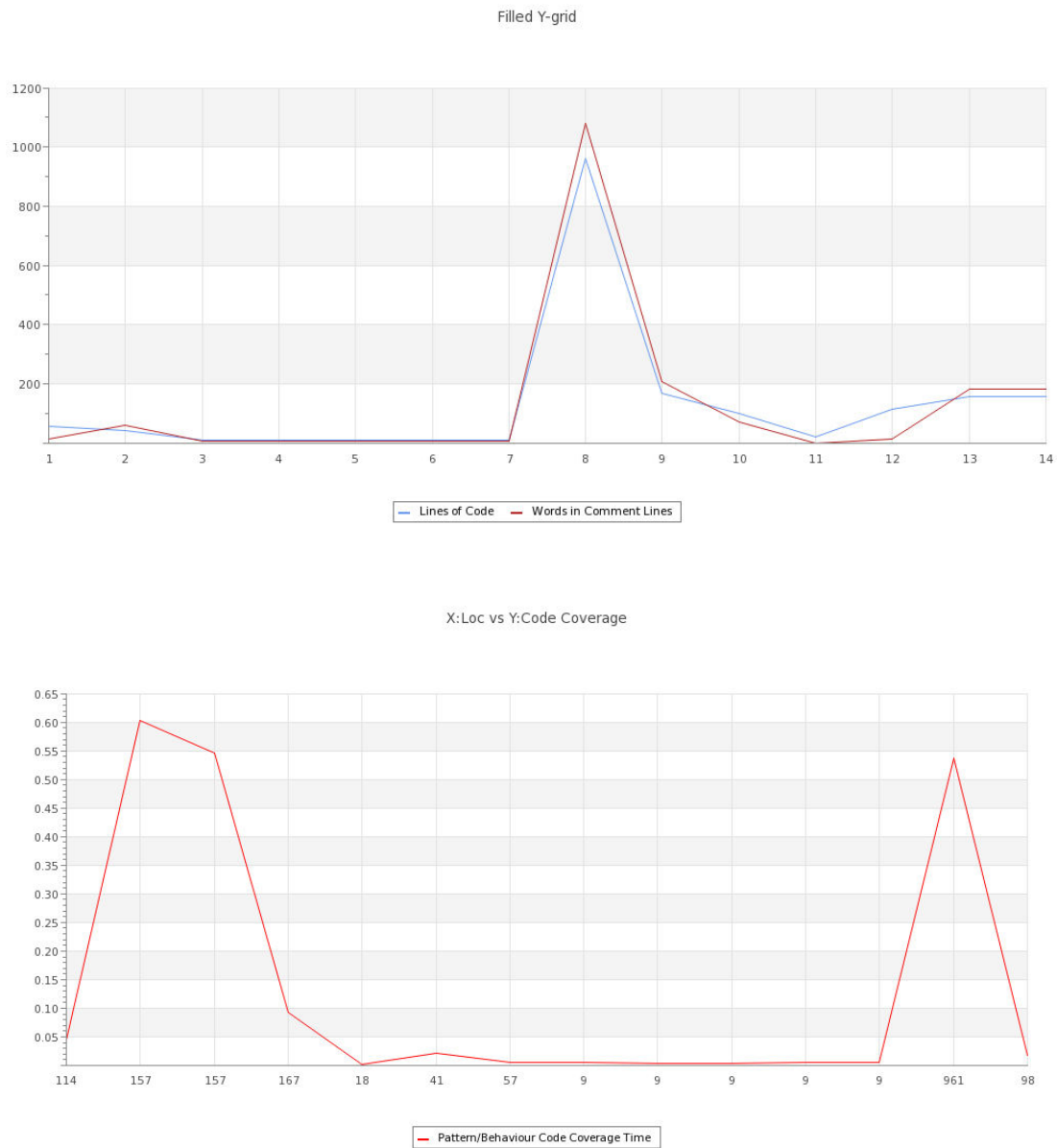
International Manuscript ID : 23482001V3I101012015-10

(Approved and Registered with Govt. of India)

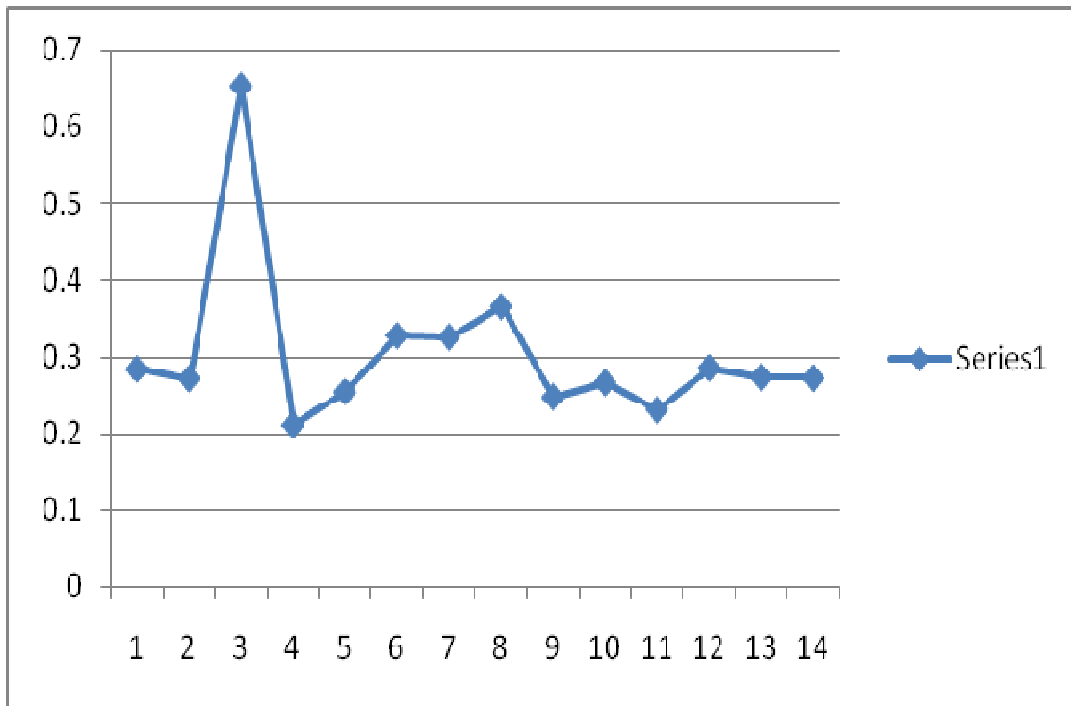
*something, sometime, sometimes, somewhere, still, such, system, take, ten, than, that, the, their, them, themselves, then, thence, there, thereafter, thereby, therefore, therein, thereupon, these, they, thick, thin, third, this, those, though, three, through, throughout, thru, thus, to, together, too, top, toward, towards, twelve, twenty, two, un, under, until, up, upon, us, very, via, was, we, well, were, what, whatever, when, whence, whenever, where, whereafter, whereas, whereby, wherein, whereupon, wherever, whether, which, while, whither, who, whoever, whole, whom, whose, why, will, with, within, without, would, yet, you, your, yours, yourself, yourselves*

Stop Words Found : QUALITY OF THE SOURCE CODE IS EXCELLENT

Filename	LOC	Comments	Words in Comments	Total Words	Comments Density	Executable LOC	Execution Time
graph.php	57	4	14	161	8.695652173913	29	0.0052249431610107
comments.php	41	13	61	125	48.8	9	0.020942211151123
a.cpp	9	1	5	20	25	3	0.0047919750213623
a.cpp	9	1	5	20	25	3	0.004608154296875
a.cpp	9	1	5	20	25	3	0.0029840469360352
a.cpp	9	1	5	20	25	3	0.0032799243927002
a.cpp	9	1	5	20	25	3	0.0047969818115234
wp-login.php	961	248	1079	4413	24.450487196918	278	0.53639197349548
wp-comments-post.php	167	58	205	658	31.155015197568	47	0.093575954437256
index.php	98	5	71	284	25	27	0.017119884490967
data.php	18	0	0	29	0	6	0.0013720989227295
clusterpurchase.php	114	5	14	693	2.020202020202	53	0.044317007064819
jpggraph_error.php	157	52	182	459	39.651416122004	52	0.54635000228882



**Figure 2 – Implementation Results**

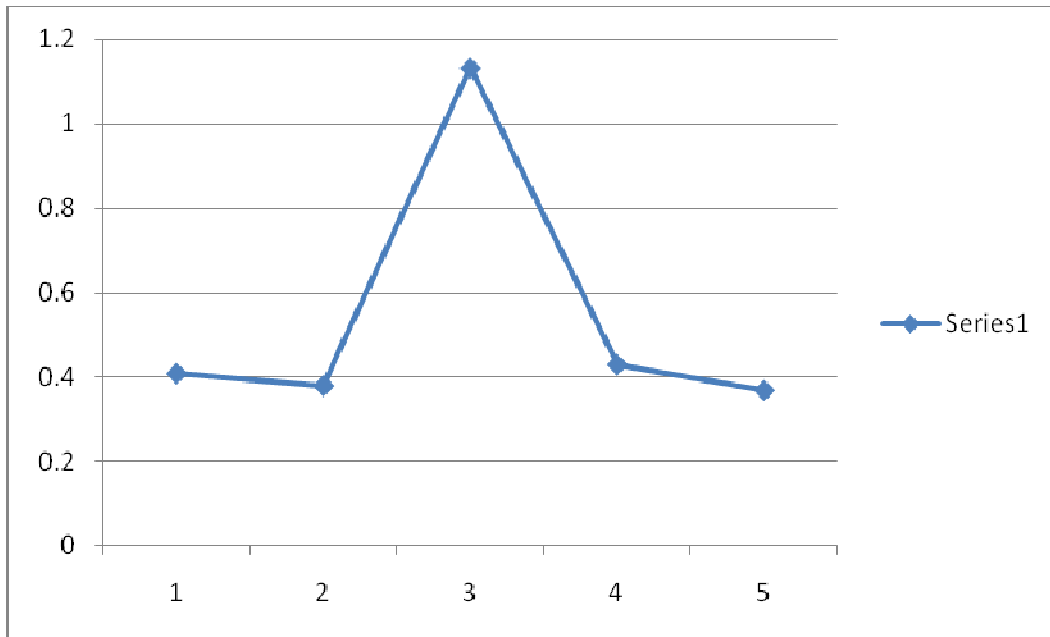


**Figure 3 – Consistency Analysis of Web Server in terms of Execution Time**

From the above graph, the performance of web server is almost consistent in terms of execution time. If number of files are 2, the average execution time of the test cases are increasing in Linear Distribution. By this aspect, the performance of server is moderate and acceptable because the web server is not behaving in abnormal way and the linear growth of output is followed.

Following is the results of simulation in terms of files in different slots of average files size.

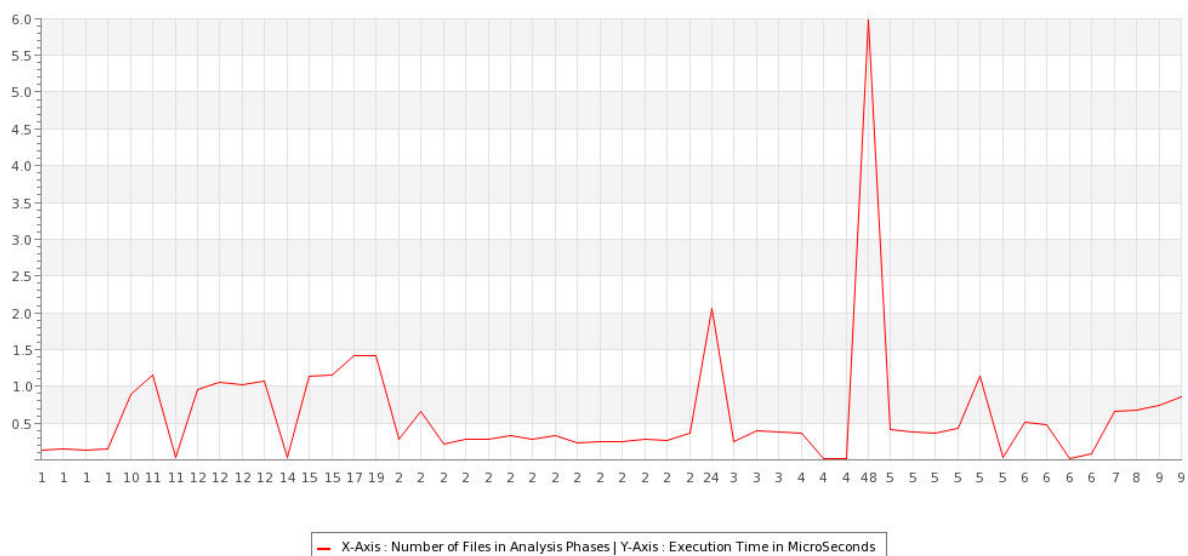
Number of files	Average File Size (In Bytes)	Average execution time
5	80.8	0.40917182
5	81.8	0.379914999
5	93.6	1.132224083
5	102.8	0.430104017
5	106.4	0.368839025

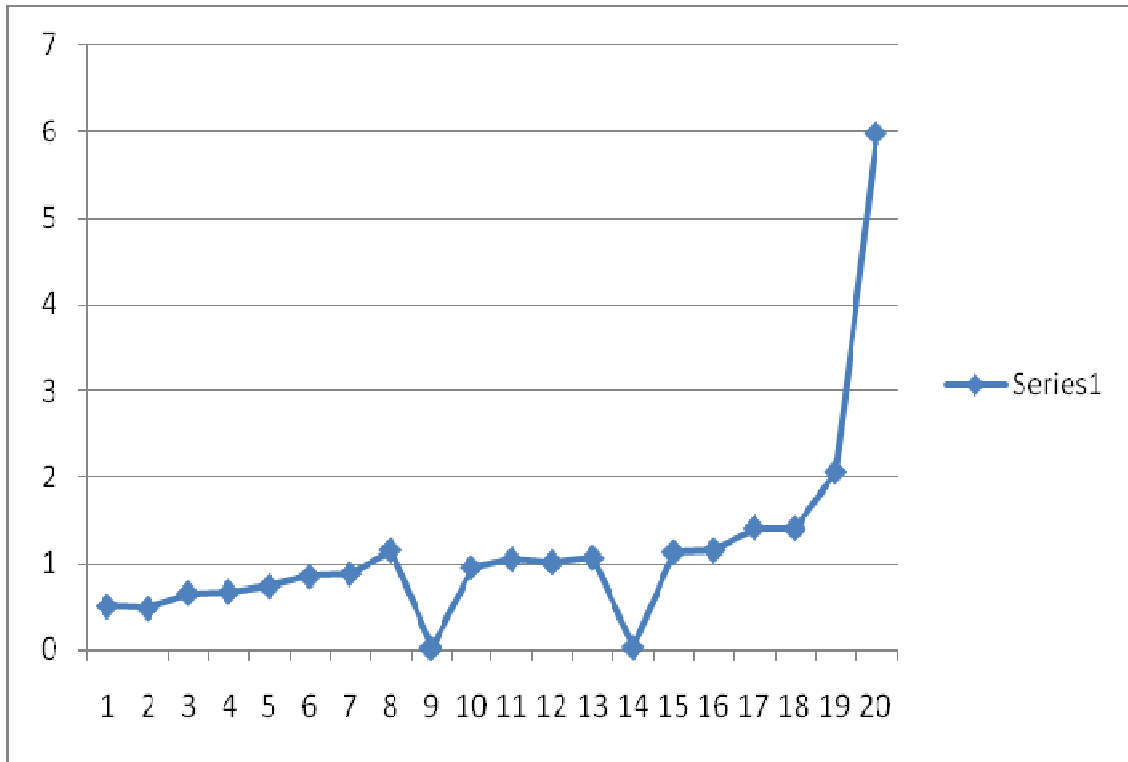


**Figure 4 – Performance Evaluation of Web Server in terms of Execution Time with Number of Simulation Attempts**

Here it is evident that performance of web server is steady in most of the cases. The server is not behaving and processing the results in abnormal way.

Line Graph of cumulative report





**Figure 5 – Performance Evaluation with File Size and Execution Time**

Now, it is clear the results are abnormal attempt 20 where the number of files is 48 and average file size is 89. In this case, the execution time is 5.97 ms which is very high as compared to the other values. It is evident from the results that the web server is having huge overhead in case of increasing number of files rather than single file with more size. The web server is able to handle less number of files in more effective way as compared to more files with lesser size.

---

## RESULTS FROM IOT BASED SERVER

Implementation of Security and Integrity in Internet of Things (IoT)



Figure 6 – Simulation Scenario of IoT Devices




		CAR	199.200.10.2	<a href="#">Select</a>
		CHILD	200.140.49.12	<a href="#">Select</a>
		WASHING MACHINE	230.43.52.11	<a href="#">Select</a>
		COMPUTER	100.242.4.23	<a href="#">Select</a>
		BIKE	100.200.49.89	<a href="#">Select</a>
		AC	200.98.84.13	<a href="#">Select</a>


Figure 7 – Simulation Scenario of IoT Devices in Connection



Figure 8 – Authentication Process Initiates

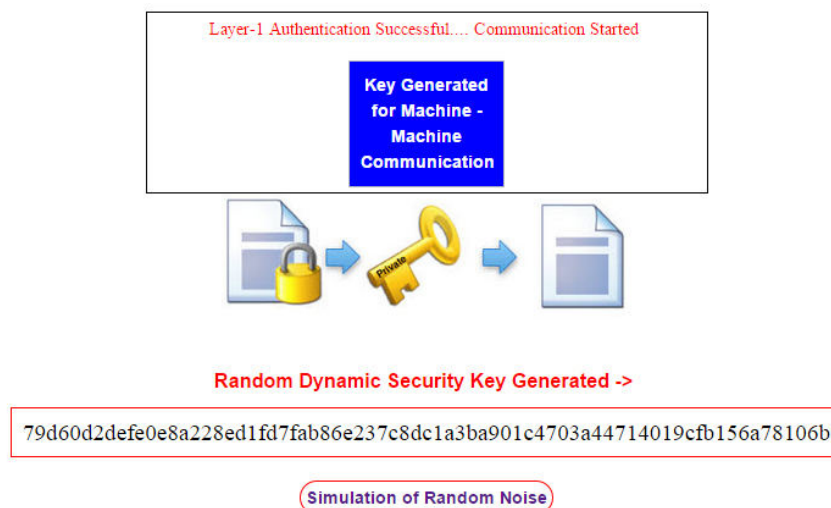
International Refereed Journal of Reviews and Research  
Volume 3 Issue 1 January-February 2015  
International Manuscript ID : 23482001V3I101012015-10  
(Approved and Registered with Govt. of India)

**Authentication Process Initialized**

  
car

Personalized Queries	
COLOR	<input type="text"/>
MODEL	<input type="text"/>
BRAND	<input type="text"/>
<input type="button" value="Authenticate"/>	

Figure 9 – Queries of Personalized Aspects



International Refereed Journal of Reviews and Research  
 Volume 3 Issue 1 January-February 2015  
 International Manuscript ID : 23482001V3I101012015-10  
 (Approved and Registered with Govt. of India)



Figure 10 – Comparison of classical and proposed approach in terms of Execution Time

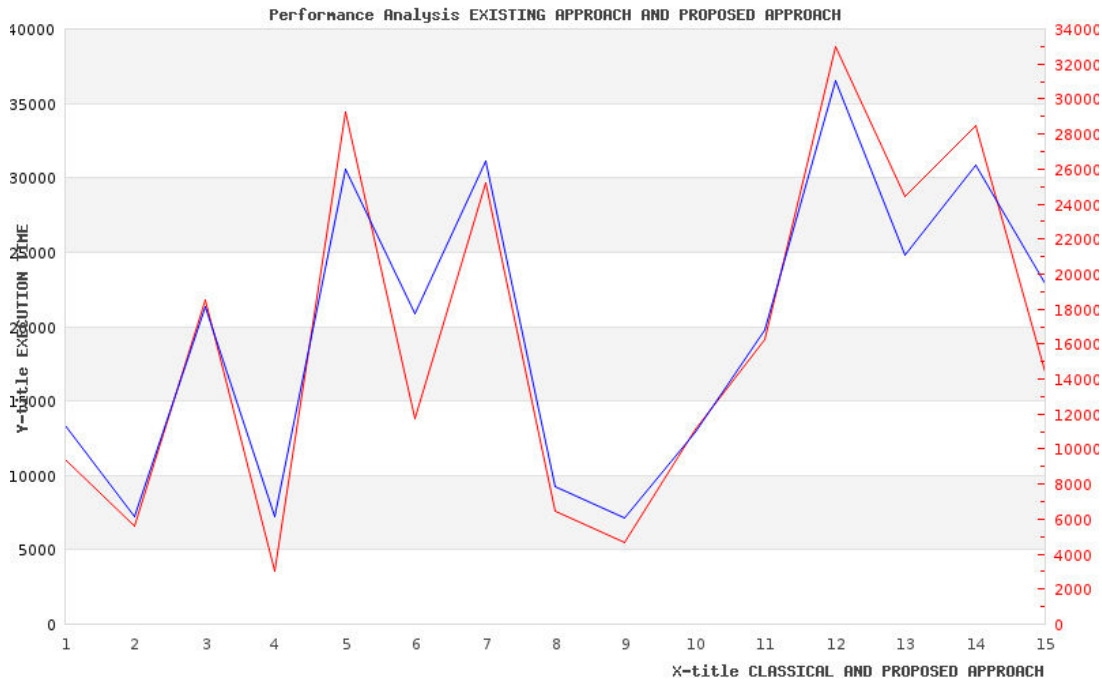


Figure 11 – Comparison of classical and proposed approach in terms of Cost

## REFERENCES

- [1] Rayadurgam, S., & Heimdahl, M. P. E. (2001). Coverage based test-case generation using model checkers. In *Engineering of Computer Based Systems*, 2001. ECBS 2001.Proceedings. Eighth Annual IEEE International Conference and Workshop on the (pp. 83-91). IEEE.
- [2] Ali, S., Briand, L. C., Hemmati, H., & Panesar-Walawege, R. K. (2010). A systematic review of the application and empirical investigation of search-based test case generation. *Software Engineering, IEEE Transactions on*, 36(6), 742-762.
- [3] Rasal, Y. M., & Nagpure, S. (2015). Web Application: Performance Testing Using Reactive Based Framework. *IJRCCT*, 4(2), 114-118.

- [4] Yan, M., Sun, H., Liu, X., Deng, T., & Wang, X. (2015). Delivering Web service load testing as a service with a global cloud. *Concurrency and Computation: Practice and Experience*, 27(3), 526-545.
- [5] Deng, Z., Li, X., Ruan, F., & Ma, W. (2015). The Designation and Application of a Load Balancing Strategy Used For Session Persistence Based on Dynamic Message Queue.
- [6] Martínez-Álvarez, A., Cuenca-Asensi, S., Ortiz, A., Calvo-Zaragoza, J., & Tejuelo, L. A. V. (2015). Tuning compilations by multi-objective optimization: Application to apache web server. *Applied Soft Computing*, 29, 461-470.
- [7] Vokorokos, L., Baláž, A., & Ádám, N. (2015). Secure Web Server System Resources Utilization. *Acta Polytechnica Hungarica*, 12(2).
- [8] Maddelein, D., Colaert, N., Buchanan, I., Hulstaert, N., Gevaert, K., & Martens, L. (2015). The iceLogo web server and SOAP service for determining protein consensus sequences. *Nucleic acids research*, gkv385.
- [9] Tracey, N., Clark, J. A., & Mander, K. (1998). The way forward for unifying dynamic test-case generation: The optimisation-based approach. In Proceedings of the IFIP International Workshop on Dependable Computing and Its Applications (DCIA).
- [10] Perrouin, G., Sen, S., Klein, J., Baudry, B., & Le Traon, Y. (2010, April). Automated and scalable t-wise test case generation strategies for software product lines. In Software Testing, Verification and Validation (ICST), 2010 Third International Conference on (pp. 459-468). IEEE.
- [11] Sant, J., Souter, A., & Greenwald, L. (2005, May). An exploration of statistical models for automated test case generation. In ACM SIGSOFT Software Engineering Notes (Vol. 30, No. 4, pp. 1-7). ACM.
- [12] Hanna, S., & Munro, M. (2007, May). An approach for specification-based test case generation for Web services. In Computer Systems and Applications, 2007.AICCSA'07. IEEE/ACS International Conference on (pp. 16-23). IEEE.

- [13] Tai, K. C., & Lie, Y. (2002). A test generation strategy for pairwise testing. *IEEE Transactions on Software Engineering*, 28(1), 109-111.
- [14] Rothermel, G., Untch, R. H., Chu, C., & Harrold, M. J. (1999). Test case prioritization: An empirical study. In *Software Maintenance, 1999.(ICSM'99) Proceedings. IEEE International Conference on* (pp. 179-188). IEEE.
- [15] Czerwinka, J. (2006, October). Pairwise testing in the real world: Practical extensions to test-case scenarios. In *Proceedings of 24th Pacific Northwest Software Quality Conference, Citeseer* (pp. 419-430).
- [16] Bertolino, A. (2007, May). Software testing research: Achievements, challenges, dreams. In *2007 Future of Software Engineering* (pp. 85-103). IEEE Computer Society.
- [17] Volume 5, Issue 1, January 2015 ISSN: 2277 128X *International Journal of Advanced Research in Computer Science and Software Engineering* Research Paper Available online at: [www.ijarcse.com](http://www.ijarcse.com) Test Case Selection and Prioritization Using Multiple Criteria M. Suppriya, A. K. Ilavarasi, Department of Computer Science Sona College of Technology, Tamil Nadu, India
- [18] Weyuker, E. J. (1998). Testing component-based software: A cautionary tale. *IEEE software*, (5), 54-59.
- [19] El-Far, I. K., & Whittaker, J. A. (2001). Model-Based Software Testing. *Encyclopedia of Software Engineering*.
- [20] Kaner, C., Bach, J., & Pettichord, B. (2008). *Lessons learned in software testing*. John Wiley & Sons.
- [21] Memon, A. M., Pollack, M. E., & Soffa, M. L. (2001). Hierarchical GUI test case generation using automated planning. *Software Engineering, IEEE Transactions on*, 27(2), 144-155.

- [22] Tung, Y. W., & Aldiwan, W. S. (2000). Automating test case generation for the new generation mission software system. In *Aerospace Conference Proceedings, 2000 IEEE* (Vol. 1, pp. 431-437). IEEE.
- [23] Bai, X., Dong, W., Tsai, W. T., & Chen, Y. (2005, October). WSDL-based automatic test case generation for web services testing. In *Service-Oriented System Engineering, 2005. SOSE 2005. IEEE International Workshop* (pp. 207-212). IEEE.
- [24] Ali, S., Briand, L. C., Hemmati, H., & Panesar-Walawege, R. K. (2010). A systematic review of the application and empirical investigation of search-based test case generation. *Software Engineering, IEEE Transactions on*, 36(6), 742-762.
- [25] Fröhlich, P., & Link, J. (2000). Automated test case generation from dynamic models. In *ECOOP 2000—Object-Oriented Programming* (pp. 472-491). Springer Berlin Heidelberg.
- [26] Mingsong, C., Xiaokang, Q., & Xuandong, L. (2006, May). Automatic test case generation for UML activity diagrams. In *Proceedings of the 2006 international workshop on Automation of software test* (pp. 2-8). ACM.
- [27] Perrouin, G., Sen, S., Klein, J., Baudry, B., & Le Traon, Y. (2010, April). Automated and scalable t-wise test case generation strategies for software product lines. In *Software Testing, Verification and Validation (ICST), 2010 Third International Conference on* (pp. 459-468). IEEE.
- [28] D. Spinellis, "Code Quality: The Open Source Perspective", Addison-Wesley, Boston - MA, 2003.
- [29] B. N. Corwin, R. L. Braddock, "Operational performance metrics in a distributed system", Symposium on Applied Computing, Missouri - USA, 1992, pp. 867-872.
- [30] R. Numbers, "Building Productivity Through Measurement", Software Testing and Quality Engineering Magazine, vol 1, 1999, pp. 42-47
- [31] IFPUG - International Function Point Users Group, online, last update: 03/2008, available: <http://www.ifpug.org/>



- [32] B. Boehm, "Cost Models for Future Software Life Cycle Processes: COCOMO 2.0", U.S.Center for Software Engineering, Amsterdam, 1995, pp. 57-94.
- [33] N. E. Fenton, M. Neil, "Software Metrics: Roadmap", International Conference on Software Engineering, Limerick - Ireland, 2000, pp. 357–370.
- [34] M. K. Daskalantonakis, "A Pratical View of Software Measurement and Implementation Experiences Within Motorola", IEEE Transactions on Software Engineering, vol 18, 1992, pp. 998–1010.
- [35] R. S. Pressman, "Software engineering a practitioner's approach", 4th.ed, McGraw-Hill, New York - USA, 1997, pp. 852.
- [36] I. Sommerville, "Engenharia de Software", Addison-Wesley, 6º Edição, São Paulo – SP, 2004.
- [37] D. C. Ince, M. J. Sheppard, "System design metrics: a review and perspective", Second IEE/BCS Conference, Liverpool - UK, 1988, pp. 23-27.
- [38] L. C. Briand, S. Morasca, V. R. Basili, "An Operational Process for Goal-Driven Definition of Measures", Software Engineering - IEEE Transactions, vol 28, 2002, pp. 1106-1125.
- [39] Refactorit tool, online, last update: 01/2008, available: <http://www.aqris.com/display/ap/RefactorIt>
- [40] O. Burn, CheckStyle, online, last update: 12/2007, available: <http://eclipse-cs.sourceforge.net/index.shtml>
- [41] M. G. Bocco, M. Piattini, C. Calero, "A Survey of Metrics for UML Class Diagrams", Journal of Object Technology 4, 2005,pp. 59-92.
- [42] JDepend tool, online, last update: 03/2006,available: <http://www.clarkware.com/software/JDepend.html>
- [43] Metrics Eclipse Plugin, online, last update: 07/2005, available: <http://sourceforge.net/projects/metrics>

- [44] Coverlipse tool, online, last update: 07/2006, available: <http://coverlipse.sourceforge.net/index.php>
- [45] JHawk Eclipse Plugin, online, last update: 03/2007, available: <http://www.virtualmachinery.com/jhawkprod.htm>
- [46] S. Morasca, L. C. Briand, V. R. Basili, E. J. Weyuker, M. V. Zelkowitz, B. Kitchenham, S. Lawrence Pfleeger, N. Fenton, "Towards a framework for software measurementvalidation", Software Engineering, IEEE Transactions, vol 23, 1995, pp. 187-189.
- [47] H. F. Li, W. K. Cheung, "An Empirical Study of Software Metrics", IEEE Transactions on Software Engineering, vol 13, 1987, pp. 697-708.
- [48] H. Zuse, "History of Software Measurement", online, last update: 09/1995, lavailable: [http://irb.cs.tu-berlin.de/~zuse/metrics/History\\_00.html](http://irb.cs.tu-berlin.de/~zuse/metrics/History_00.html)
- [49] N. E. Fenton, M. Neil, "Software Metrics: Roadmap", International Conference on Software Engineering, Limerick - Ireland, 2005, pp. 357–370.
- [50] T. J. McCabe, "A Complexity Measure". IEEE Transactions of Software Engineering, vol SE-2, 1976, pp. 308-320.
- [51] D. Kafura, G. Reddy, "The Use of Software Complexity Metrics in Software Maintenance", IEEE Transactions on Software Engineering archive, vol 13 , New Jersey - USA, 1987, pp. 335-343.
- [52] B. Ramamurty, A. Melton, "A Syntheses of Software Science Measure and The Cyclomatic Number", IEEE Transactions on Software Engineering, vol 14, New Jersey - USA, 1988, pp. 1116-1121.
- [53] J. K. Navlakha, "A Survey of System Complexity Metrics", The Computer Journal, vol 30, Oxford - UK, 1987, pp. 233-238.
- [54] E. VanDoren, K. Sciences, C. Springs, "Cyclomatic Complexity", online, last update: 01/2007, available: [http://www.sei.cmu.edu/str/descriptions/cyclomatic\\_body.html](http://www.sei.cmu.edu/str/descriptions/cyclomatic_body.html)

**ISSN (Online) : 2348 - 2001**

International Refereed Journal of Reviews and Research

Volume 3 Issue 1 January-February 2015

International Manuscript ID : 23482001V3I101012015-10

(Approved and Registered with Govt. of India)