IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-2, Issue-9 E-ISSN: 2347-2693

Dynamic Fault Localization in Web Application

Vijay Kumar .Akula[1] and Chaitanya Kumar .N[2*]

www.ijcaonline.org

Received: Aug/26/2014 Revised: Sep/13/2014 Accepted: Sep/26/2014 Published: Sep/30/2014

***Abstract*—** Now-a-days, web applications have become prevalent around the world. This technology has made it possible to carry on business along the web. Many companies developing web applications. There is a need of locating faults in web applications. In this report, we present a technique to locate faults in web applications. Dynamic Fault Localization is a process to localize the faulty statements in web application programs. To locate faulty statement in web programs we implement Tarantula, Ochiai, and Jaccard Coefficients.

***Keywords*—**The word Coefficient, Suspiciousness and Prediction are one are the same, fault, faulty statement, malformed statement

I. INTRODUCTION

Web application development has provided an efficient and effective route to deal with foreign clients, aimed at a purely gain in cost cutting methods. Through the deployment of customized and specified web applications, a diverse range of purpose can be served. Web applications or computer programs assist you to gather information, collect data and to find out anything that you really want. Web applications, forms on-line trading in online retail, have become prevalent because of their convenient and inexpensive ways to provide product information and service on-line to anyone at any time. As more business relies on web applications for their mission-critical operations, the quality of the applications become crucial. A small, malformed statement in the web applications could interrupt entire business on the web. There is a need to locate the faulty statement in the web application. Web applications are combinations of multiple computer programming language (like Html, JSP, Java etc.…,).

Finding and Locating the malformed statement in the web application programs is complex task [1]. Our attempt is to provide a single roof to perform tests on web application. In this paper our approach is to locate malformed statements in HTML, JSP pages [2] and locate faults in associated Java classes to run web application properly.

To cope with the preceding situation, researchers have proposed various techniques for fault localization [3, 4] to find fault exactly where is. A fault is nothing but the bug. It is always challenging for software engineers to remove bugs effectively and efficiently. To get rid of bugs, software engineers must first able to identify precisely where the bugs are, which is known as Dynamic Fault Localization.

II. RELATED WORK

JSP pages have been widely applied in Java-based Web applications to handle HTTP request, to interact with Java Components like Java Beans, and to generate dynamic pages. It is important to ensure that the JSP pages are written correctly and their interactions with other components are handled properly. However, JSP pages usually mix up scripts (i.e.., JSP scriptlet) with HTML statements in order to generate dynamic pages. This makes JSP pages difficult to understand and test. Our goal is to find the root cause for the web application failure.

A. *Html and JSP Technologies*

While contemporary Web browsers do an increasingly good job of parsing even the worst HTML & JPS “tag soup”, some errors are not always caught gracefully. Very often, different software on different platforms will not handle errors in a similar faction, making it extremely difficult to apply style or layout consistently

B. *Study of Various Methods for Fault Localization*

Fault Localization is an intensively studied research topic in regression testing. Techniques for fault localization aim to improve the rate of fault detection. In the literature, there are several lines of research on fault localization. Some are:-

Pan and Spafford [7] produced a set of heuristics that identify a set of suspicious statements in a program, but provides no ranking of the propitiousness of the statements in the bent. Therefore, the user has no information about where to start the debugging process within the set of instructions.

In addition, several of their heuristics require setting thresholds that must be tuned to the particular program and fault, which is not recognized until the faults are found. As well, their approach does not offer a means for the user to interface with the provided information for debugging.

Program slicing is a commonly employed technique for debugging. A statement program [10] slice for a give variable at a given statement contains all the executable statement that could possibly affect the value of this variable at the statement. Reduction of the debugging search domain via slicing is based on the idea that is a test case fails due to an incorrect variable value at a statement then the defect should be found in the static slice association with that variable-statement pair. We

Corresponding Author: *Chaitanya Kumar .N, n*_chaitu@yahoo.com

can therefore confine our search to the slice rather than looking at the entire program.

In Program Spectrum-based Method. A Program spectrum records the execution information about a program in certain aspects, such as execution information for conditional branches or loop-free intra-procedural path [9]. It can be utilized to track program behavior [8]. When the execution fails, such information can be used to identify suspicious code that is responsible for the failure.

A Set Union and Set Intersection Method obtain set of program lines by removing the statements executed by the passed test case. This resulting set of statements is then used as the initial set of suspicions statements when searching for defects. Pan present a set of dynamic-slicing-based heuristics that use set algebra of test cases' dynamic slices for similar purpose [7]. In addition to utilizing the information obtained about, they also used analysis information to compute dynamic slices of the program and test examples from a particular program level.

In this report, we address the problems of locating faulty statements in web application development using JSP. This advance facilitates the web application developers to identify the faults and reduce his testing time and it also reduces, the load on the web server.

III. COEFFICIENT FOR FAULT LOCALIZATION

A. *Java Server Page*

In our report, we address the problems of locating faulty statements in web application development using JSP. This approach helps the web application developers to identify the errors and reduce his testing time and it also reduces the burden on the web server.

A JSP page is translated into a Java servlet before being executed. The JSP translator is typically triggered by the .jsp file name extension. JSP Translator depends on the jsp tags, the misplace of jsp tags makes the faulty servlet.

```
1.<html>
2.<body>
3.<% out.print("welcome to jsp"); %>  } JSP tag
4.</body>
5.</html>
```

Fig. 1 JSP Sample Program

B. *Hyper Text Markup Language*

HTML is the standard mark-up language used for making web pages. HTML is written in the form of HTML elements consisting of tags enclosed in angle brackets like <html>. HTML tags most commonly come in pairs, although some are unpaired tags. Using the correct HTML document structure when creating a web page is important. If the HTML document structure is incorrect the web page can break or the web browser may not be able to read the page and display to the users. Therefore, HTML validator is a program utilized to locate the unstructured tags of the HTML page, reports are generated.

C. *Coefficient Equations for Fault Localization*

Jones [11, 12] presented Tarantula, a fault localization technique that associates with each statement a suspiciousness rating that indicates the likelihood for that statement to contribute to a failure. The suspiciousness rate for individual statement is computed with the following coefficient.

$$suspiciousness_{Tarantula}^{(statement)} = \frac{\frac{failed(statement)}{totalfailed(statements)}}{\frac{passed(statement)}{totalpassed(statements)} + \frac{failed(statement)}{totalfailed(statements)}}$$

Eq. 1 Tarantula Coefficient Equation

In the area of data clustering, another similarity coefficients have been proposed that can also be used to calculate suspiciousness ratings. These include the Jaccard [13] coefficient used in the pinpoint program [14]

$$suspiciousness_{Jaccard}^{(statement)} = \frac{failed(statement)}{totalfailed(statements) + passed(statement)}$$

Eq. 2 Jaccard Coefficient Equation.

Ochiai [15] Coefficient equation, it is form molecular biology domain. We compute the coefficients (i.e.., suspiciousness) for individual statements of the program. In the equations, passed(statement) and failed(statement) represents the number of passing test cases and the number of failing test cases that executes the statement of the program tested. The term totalfailed(statements) denotes the total number of failing test cases.

$$suspiciousness_{Ochiai}^{(statement)} = \frac{failed(statement)}{\sqrt{totalfailed(statements) * (passed(statement) + failed(statement)}}$$

Eq. 3 Ochiai Coefficient Equation.

Ranking the statements based on the suspiciousness rating have been computed. These ranks need not be unique, the rank of each statement is the number of statements with greater than or equal suspiciousness:

$$rank(statement) = |statement' : suspiciousness(statement') \geq suspiciousness(statement)|$$

Eq. 4 Compute Rank

The rank of statement(s) reflects the maximum number of statements that would have to be examined if statements are examined in order of decreasing suspiciousness, and if 1 were the last statement of that particular suspiciousness level chosen for examination.

IV. OUR APPROACH

Dynamic Fault Localization is an approach address failure of web application programs. In "Dynamic Fault Localization" the user login into the scheme, and utilizes the services of the Dynamic Fault Localization. The Services of the Dynamic Fault localization are: (a) Html Validation (b) JSP Validation and (c) Locate Faults in JSP program and its associated Java Class. The figure 2 depicts the operation flow of the system.

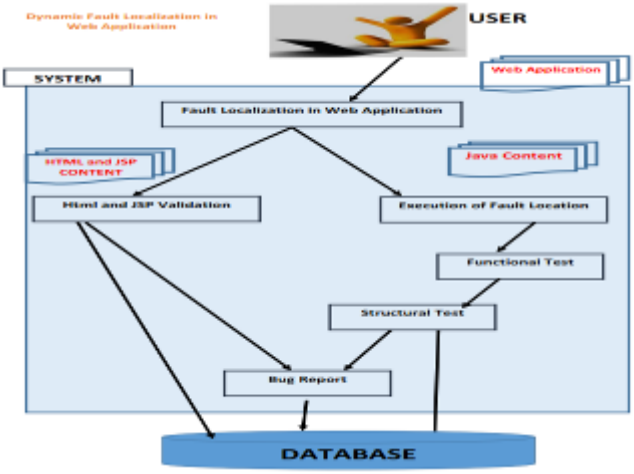

Fig. 2 Process flow diagram

A. *Calculating Coefficientds using Equations*

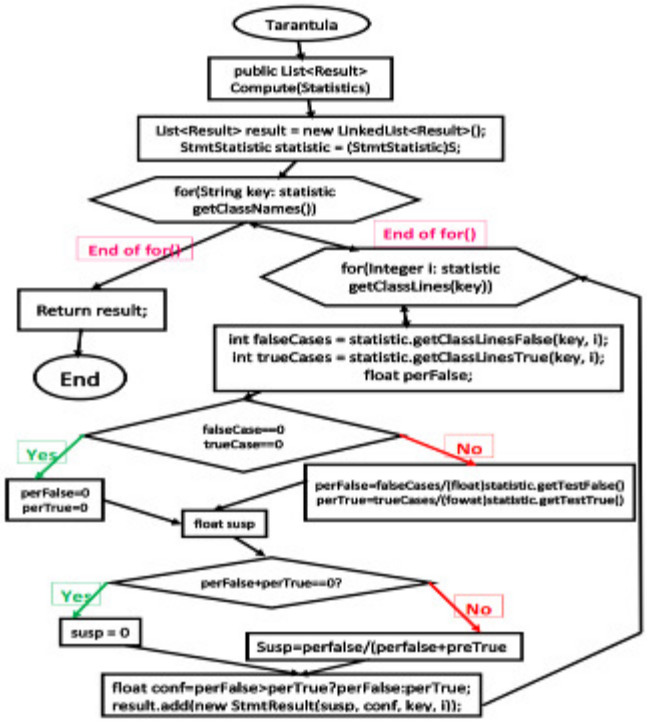

Fig. 3 Tarantula Coefficient Flowchart

In this section, we implement programs to locate faults in the web applications, it detects the faults in the JSP program and its associated Java classes. And so, the developer needs to modify the faulty statements localized by the organization.

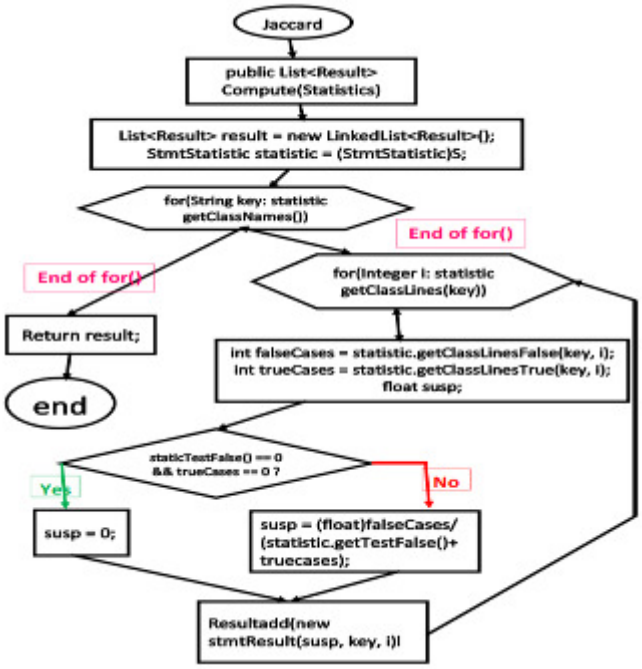

Fig. 4 Jaccard Coefficient Flowchart

These processes involve two steps. First, Fault detection is to check whether the set of lines of code is fault free or not. Second, Fault localization is to locate the faulty statement position and The figure 3, 4 and 5 are the flow chart of fault localization coefficients. Which are implemented into our System.

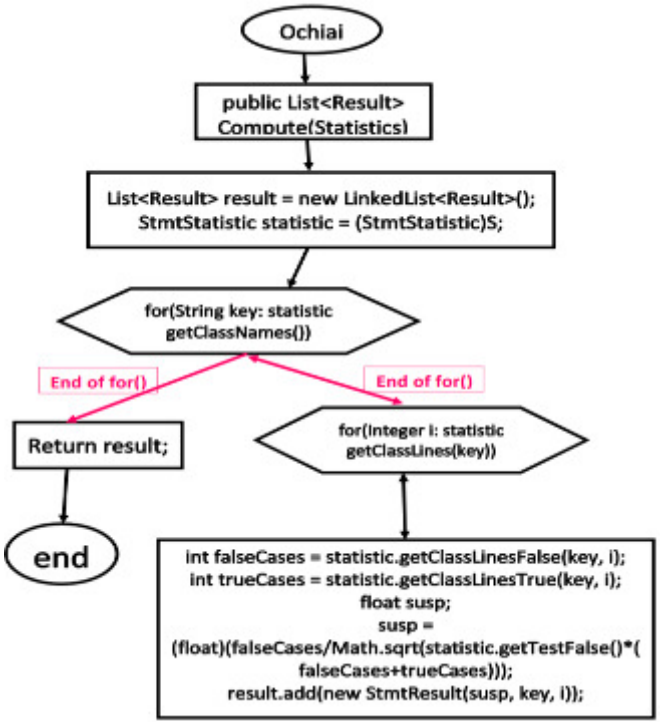

Fig. 5 Ochiai Coefficient Flowchart

Fault localization is applied for both detecting errors and locating faults. The system implements the fault localization

equations defined in the section 3 (i.e.., Tarantula, Jaccard and Ochiai coefficient equations). These equations are used in localizing the position of the faulty statement. In the system the programmer checks for the faults in the program. Then the system generates reports. The reports look like fig 7, 8 and 9.If any malformed set of lines in the program (i.e.., that means the program contains faults). Then, the programmer localizing position of the faulty statement in the program. The localization report will provide the information about the malformed statements in the program. The figure 3 is a Tarantula coefficient implemented flowchart depicts the process of localizing the faulty statements in the program.

All the fault localization techniques use the percentage of failing and passing tests executing a given statement to calculate the statement's suspiciousness score. To this end, our system records the set of executing statement, suspiciousness, statement line number and rank of suspiciousness for each execution. Jaccard and Ochiai coefficient also implement in the system programmatically in a similar manner to that of Tarantula. The figure 4 and figure 5 is a Jaccard and Ochiai Coefficient flow charts respectively.

B. Localization of Faults in Html and JSP Web Pages

The HTML and JSP Validation (figure 6) helps in checking the validity of I-documents (Internet documents). Most of dynamic web applications are written in JSP and its response I-documents (Internet documents) are in Hypertext Mark-up Language. Mark-up language specification embodies a machine-readable formal synchronic linguistic vocabulary. The act to check a document against these constraints of markup language specification is called as Validation. The HTML and JSP Validation, validate internet documents written in HTML. The figure 6 is a flow diagram depicts the process of HTML and JSP Validation.

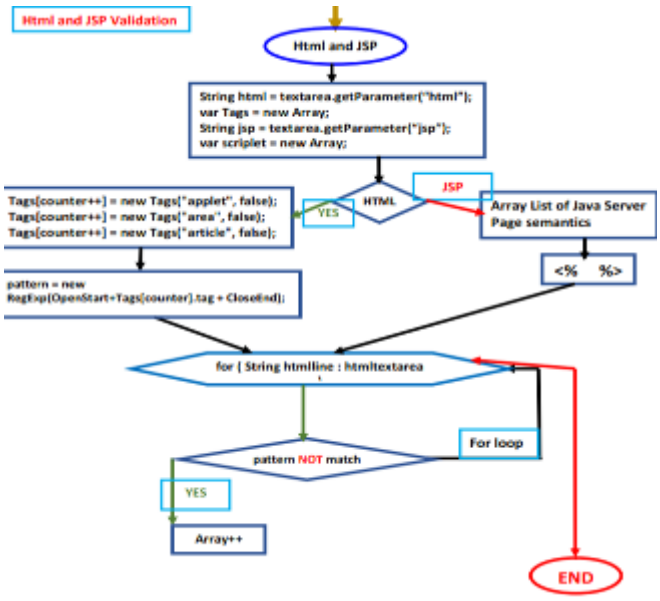

Fig. 6 Html and JSP Validation

	rank	suspicious	line	class
	1	1.0 [1.0]	10	com.test.SourceCode
	2	0.8333333 [1.0]	9	com.test.SourceCode
	3	0.71428573 [1.0]	7	com.test.SourceCode
	7	0.5 [1.0]	6	com.test.SourceCode
	7	0.5 [1.0]	5	com.test.SourceCode
	7	0.5 [1.0]	3	com.test.SourceCode
	7	0.5 [1.0]	17	com.test.SourceCode
	8	0.0 [0.6]	12	com.test.SourceCode
	9	0.0 [0.4]	14	com.test.SourceCode
	12	0.0 [0.2]	15	com.test.SourceCode
	12	0.0 [0.2]	13	com.test.SourceCode
	12	0.0 [0.2]	8	com.test.SourceCode

Fig. 7 Tarantula Report

Fault localization system records the results of all their results of fault check condition in the executed program. It records the observation consisting of the statement's line number, code file name, suspiciousness and rank are calculated using equation 4.

	rank	suspicious	line	class
	1	1.0	10	com.test.SourceCode
	2	0.5	9	com.test.SourceCode
	3	0.33333334	7	com.test.SourceCode
	7	0.16666667	6	com.test.SourceCode
	7	0.16666667	5	com.test.SourceCode
	7	0.16666667	3	com.test.SourceCode
	7	0.16666667	17	com.test.SourceCode
	12	0.0	15	com.test.SourceCode
	12	0.0	14	com.test.SourceCode
	12	0.0	13	com.test.SourceCode
	12	0.0	12	com.test.SourceCode
	12	0.0	8	com.test.SourceCode

Fig. 8 Jaccard Report

The recorded observations are viewed in tabular form, suspiciousness rate and its line number, thus the programmer fine the faulty statement location.

	rank	suspicious	line	class
	1	1.0	10	com.test.SourceCode
	2	0.70710677	9	com.test.SourceCode
	3	0.57735026	7	com.test.SourceCode
	7	0.4082483	6	com.test.SourceCode
	7	0.4082483	5	com.test.SourceCode
	7	0.4082483	3	com.test.SourceCode
	7	0.4082483	17	com.test.SourceCode
	12	0.0	15	com.test.SourceCode
	12	0.0	14	com.test.SourceCode
	12	0.0	13	com.test.SourceCode
	12	0.0	12	com.test.SourceCode
	12	0.0	8	com.test.SourceCode

Fig. 9 Ochiai Report

V. EXPERIMENTAL RESULT

In our experimentation, we examine the effectiveness of locating faults and the program under examination can be assessed by counting and classifies discover of faults.

The figure 7, figure 8 and figure 9 are the report generated using tarantula, Jaccard and Ochiai coefficients respectively to localize faulty statement.

The following figures are graphs plotted using the reports generated by the tarantula, Jaccard and Ochiai respectively.

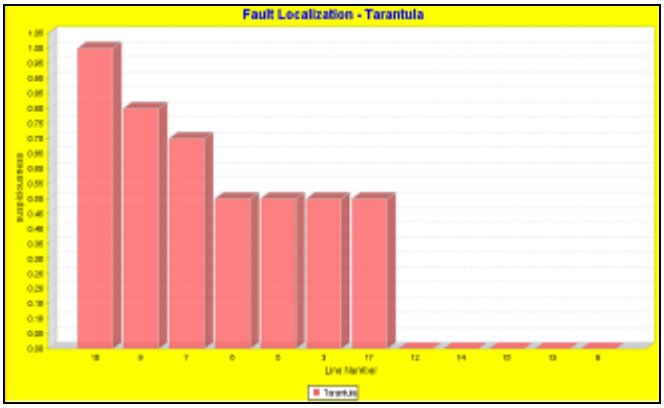

Fig. 10 Tarantula bar graph

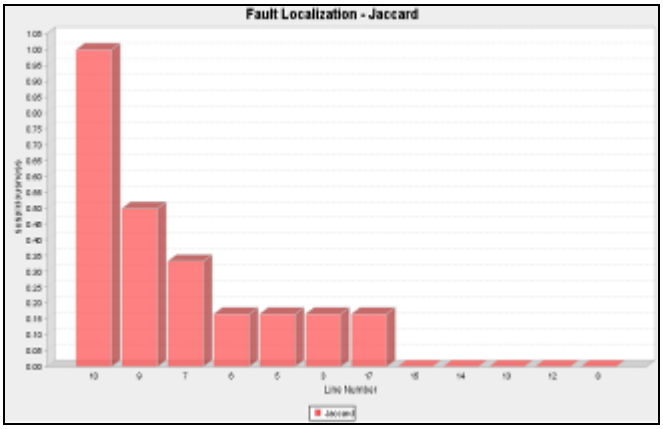

Fig. 11 Jaccard bar diagram

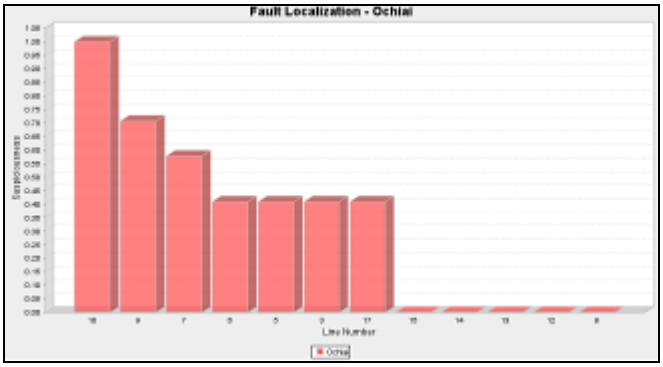

Fig. 12 Ochiai Report

From the figure 7 tarantula report. We can see that the statement on the line number 10 is faulty statement. Which has highest suspicious percentage rate and it is rated as 1. The fig 10 Tarantula bar graph plotted using the suspiciousness rate and statement line number of the tarantula report.

A. *Comparative Graph*

The evaluation of the above coefficients is shown in the below graph

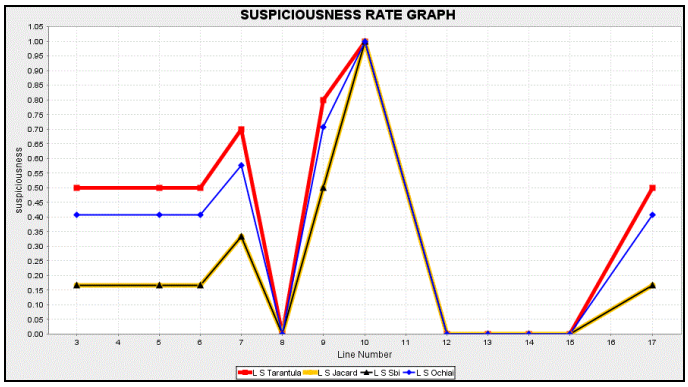

Fig. 13 Comparative Graph

We observe that the Tarantula coefficient localize faults 87.8 percent of the fault within 1 percent of all executed statements followed by Ochiai 78.11 percentage and next Jaccard 69.23 percent.

VI. CONCLUSION

After completing the research tasks that we outlined in the paper, we expect to make the following contributions in the dissertation:

- An approach to locate faults in web applications.
- Implementation of the approach in a tool
- A comparison of different fault localization algorithm results
- More insight into what relationships of the passing and the failing test cases affect the effectiveness of fault localization
- Execution of the overture in a creature

We observed the more than 80% of all faults in the program are found and located with 1% of all executed statements of the program.

REFERENCES

[1] Analysing and Testing Web Application Performance Research Inventy: International Journal of Engineering and Science Vol.3, Issue 10 (October 2013), PP 47-50

[2] Improving Browsing Environment Compliance Evaluations for Websites Cyntrica Eaton 4166 A. V. Williams Building Department of Computer Science University of Maryland,

[3] Bo Jiang, Zhenyu Zhang, T.H. Tse, T.Y. Chen, "How Well Do Test Case Prioritization Technique Support Statistical Fault Localization", International Computer Software and Applications Conference, 2009.

[4] Fault Localization for Dynamic Web Applications, IEEE Transactions On Software Engineering, Vol. 38, No. 2, March/April 2012.

[5] Eric Wong, Vidroha Debroy, " Software Fault Localization" W. IEEE Annual Technology Report, 2009.

[6] http://www.w3.org/TR/html4/

[7] H. Pan and E. H. Spafford. Heuristics for automatic localization of software faults. Technical Report SERC-TR-116-P, Purdue University, July 1992.

[8] T. Reps, T. Ball, M. Das, and J. Larus, "The Use of Program Profiling for Software Maintenance with Applications to the Year 2000 Problem," in Proceedings of the 6th European Software Engineering Conference, pp. 432-449, Zurich, Switzerland, September, 1997

[9] M. J. Harrold, G. Rothermel, K. Sayre, R. Wu, and L. Yi, "An Empirical Investigation of the Relationship between Spectra Differences and Regression Faults," Journal of Software Testing, Verification and Reliability, 10(3):171-194, September 2000

[10] M. Weiser, "Program slicing," IEEE Transactions on Software Engineering, SE-10(4):352-357, July 1984

[11] J.A. Jones and M.J. Harrold, "Empirical Evaluation of the Tarantula Automatic Fault-Localization Technique," Proc. IEEE/ ACM Int'l Conf. Automated Software Eng., pp. 273-282, 2005.

[12] J.A. Jones, M.J. Harrold, and J. Stasko, "Visualization of Test Information to Assist Fault Localization," Proc. Int'l Conf. Software Eng., pp. 467-477, 2002.

[13] A.K. Jain and R.C. Dubes, Algorithms for Clustering Data. Prentice- Hall, Inc., 1988.

[14] M.Y. Chen, E. Kiciman, E. Fratkin, A. Fox, and E. Brewer, "Pinpoint: Problem Determination in Large, Dynamic Internet Services," Proc. Int'l Conf. Dependable Systems and Networks, pp. 595-604, 2002

AUTHORS PROFILE

Vijay kumar Akula completed B.Tech in Nishitha College of Engineering and Technology, Hyderabad in the year 2011. Now, He is persuing Mtech degree in Sreenidhi Institue of Science and technology, Hyderabad, Telaganal.