Software Health Indicator

Suhani Gupta(suhani.g20@gmail.com)
Pranoti Gorkhe(pranotigorkhe03@yahoo.com)
Vrushali Walse(walsevrushali@gmail.com)
Trupti Tarse(trupti.tarse@gmail.com)
[Sinhgad Institute Of Technology, Lonavala, Pune University]

***Abstract*- Static analysis examines code in the absence of input data and without running the code, and can detect potential security violations (e.g., SQL injection), runtime errors (e.g., dereferencing a null pointer) and logical inconsistencies (e.g., a conditional test that cannot possibly be true). We describe FindBugs, an open source static analysis tool for Java, and experience using it in production settings. The proposed system maintains bug history which stores all the details from bug origin to bug resolution. Our system provides the searching based on status, priority, and operating system with fully authenticated password encryption. Also we are providing it a web interface which shows the results according to the user's designation. Thus, we are reducing the dependency on code reviewer and automate the process.**

***Keywords*- Static Analysis, Findbugs,SHI.**

I. INTRODUCTION

According to the recent research of Stanford Group only 15% of I.T. projects are successful remaining 55% are huge failure and remaining 30% are on hold because of quality reasons like budgets, bad code quality, bad performance etc.

The bad code quality causes bad performance of project. This results into the unhappiness of the client and if client is unhappy then the company will be unhappy. If the company becomes unhappy then it naturally affects the developer progress so the code quality is the viscous cycle.

To check code quality each company has a code reviewer. He should be more experienced as well as comfortable with programming language and for that company has to pay him a huge amount of money. Hence to reduce the dependency on code reviewer and automate the process we are trying to implement Software Health Indicator (SHI).

A. Static Analysis

Static code analysis is about analyzing your source code without executing them to find potential vulnerability and other code quality problems, necessary for production quality code.

- Using static code analysis in your project gives thorough analysis of your code, without executing them. Static analysis scans ALL code. If there are vulnerabilities in the distant corner of your applications which are not even used then also static analysis has a higher probability of finding those vulnerabilities.
- Second benefit of using static code analysis is you can define your project specific rules, and they will be ensured to follow without any manual intervention. If any team member forgets to follow those rules, they will be highlighted by static code analyser like findbugs.
- Third major benefit of static code analysis is they can find the bug early in development cycle which means less cost to fix them. All these advantage of static code analyser can be best utilized only if they are part of build process.

B. Findbugs

Findbugs is a static analysis tool that examines class or jar files looking for potential problems by matching byte codes against a list of bug patterns. Findbug uses a series of ad-hoc techniques designed to balance precision, efficiency and usability. One of the main techniques findbugs uses is to syntactically match source code to known suspicious programming practices. [1]

II. RELATED WORK

Many techniques have been developed over the years to automatically find bugs in software. While these techniques are valuable, they can be difficult to apply, they are not always effective in finding real bugs. Bug patterns are code idioms that are often errors. Hence, bug pattern detectors were used to find real bugs in several real-world java applications and libraries [2]. But the bug patterns which are not defined in bug detectors will remain undetected.

A case study was done using the combination of bug finding tools with reviews and tests can be used to find bugs[3]. This automated static analysis will reduce the time spend for testing and also saves cost. But this can be applied only when the number of false positive are low. An article was proposed by Almazan[1] in which a meta-tool was used that combined the output of various static analysis tools together looking for particular lines of code, methods and classes that many tools warn about. But it's difficult to implement because of the quantity of output.

A case study by Valente[4] on the relevance of the warnings reported by bug finding tools (Findbugs and PMD) was done. It was found that by using FindBug tool, the number of non-relevant warnings or false positives reported is reduced by a considerable amount. Ayewah[5]describes how FindBug tool evaluates defects and how it is incorporated into software development process. In our paper, we have focused on using FindBug tool for static analysis for javaapplications and libraries. We are using latest version of FindBug(2.0.3) which covers almost all bug patterns. We are also providing graphical user interface and database connectivity which stores all the bug information. And the reports will be given according to the user's designation.

III. SYSTEM ARCHITECTURE

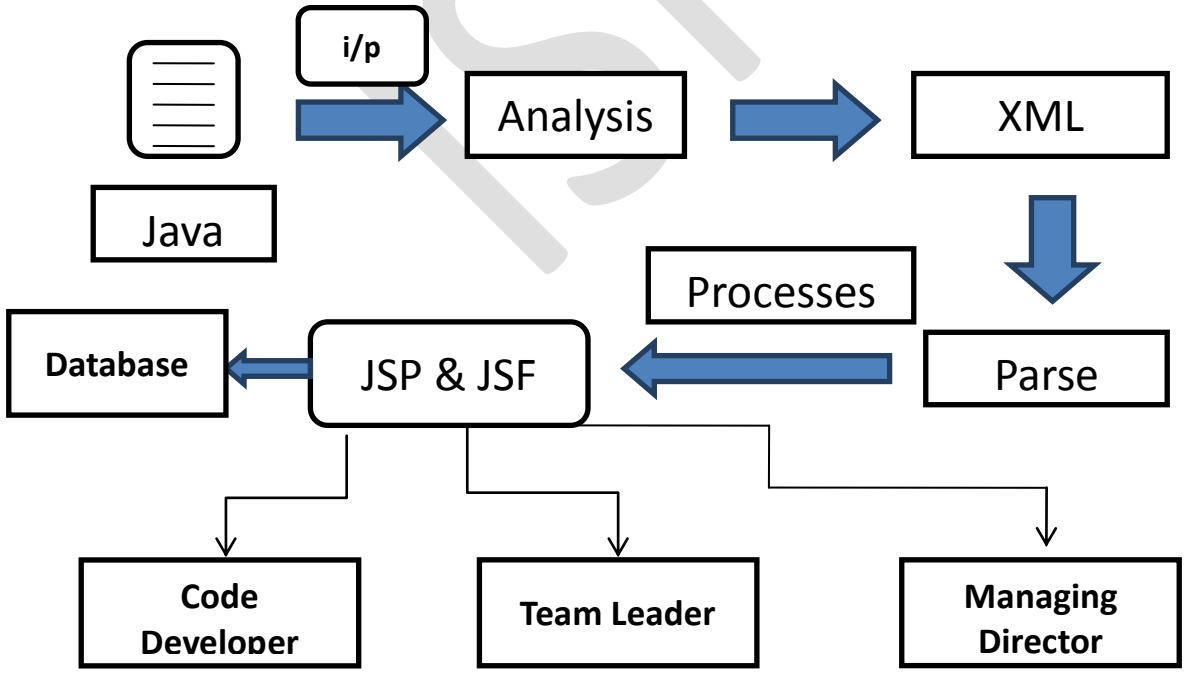

Fig. System Architecture

The above block diagram of SHI shows the basic processes that are necessary for the implementation of SHI. Stepwise processing is as follows:

(1) Step I:The java code is given as an input for the SHI.The given java code undergoes analysis phase with the help of the findbugs tool which uses Apache Ant.Apache Ant is a software tool for automating software build processes. The findbug tool produces analysis output in the xml format. Xml is a standard database

structure for the most of the information, but we can't use xml code to produce the report that has to be generated.Ant uses xml to build process and its dependencies.Thus, after analysis output is in the form of xml.

(2) Step II: To convert the xml code to java objects we use a JAXB (Java Architecture for Xml Binding). JAXB produces java objects which can be used to generate and store the historical information in the database. The reports can be shown to the multiusers like code developer, team leader and managing director as per the requirements.

(3)Step III: Java Server Pages(JSP) technology allows you to easily create Web content that has both static and dynamic components. JSP technology projects all the dynamic capabilities of Java Servlet technology but provides a more natural approach to creating static content.With the help of JSP & JSF the reports are stored in database and they are displayed according to the user's designation.

IV. ADVANTAGES

- A developer can judge its own code quality periodically.
- Project manager can take a major decision to projects which are below quality threshold.
- Team can give a standard code quality report to the client to ensure the product quality.
- No need to employ any specialized and highly paid code reviewer.
- The bugs and errors are prevented in early stage of development.

V. FUTURE MODIFICATIONS AND ENHANCEMENTS

- It could be implemented for all the programming languages.
- It could be used for instant analysis and debugging of complex code.

VI. CONCLUSIONS

An interface which examines the code without running it and can detect the bugs with the help of FindBugs tool. It maintains bug history which stores all the bug details and also gives results according to the user's designation.

ACKNOWLEDEGEMENT

We express our hearty gratitude towards our guide Prof. D. A. Lokare, Head of Computer Department Prof. T. J. Parvat and Principal Dr. M. S. Gaikwad of SIT, Lonavala for guiding us to understand the work conceptually and also for their constant encouragement.

REFERENCES

[1] Rutar, N., Almazan, C.B., Foster, J.S.: 'A comparison of bug finding tools for Java'. 15th Int. Symp.on Software Reliability Engineering (ISSRE), 2004.

[2] Hovemeyer, D., Pugh, W.: 'Finding bugs is easy', SIGPLAN Notices, 2004

[3] Wagner, S., Ju¨rjens, J., Koller, C., Trischberger, P.: 'Comparing bug finding tools with reviews and tests'. 17th Int. Conf. Testing of Communicating Systems (TestCom), 2005

[4] J.E.M. Arau´ jo, S. Souza, M.T. Valente: 'Study on the relevance of the warnings reported by Java bug-finding tools', IET Software, 2011

[5] Ayewah, N., Hovemeyer, D., Morgenthaler, J.D., Penix, J., Pugh, W.:'Using static analysis to find bugs', IEEE Softw., 2008

[6] www.findbugs.sourceforge.net