**Literature Survey**

**On**

**Diff set Based Automation**

**SUBMITTED BY**

**PRAVEENGOUDA S P 1MS12CS076**



**M. S. Ramaiah Institute of Technology**

**(Autonomous Institute, Affiliated to VTU)**

**BANGALORE-560054**

**Department of Computer Science & Engineering**

***Under the guidance of***

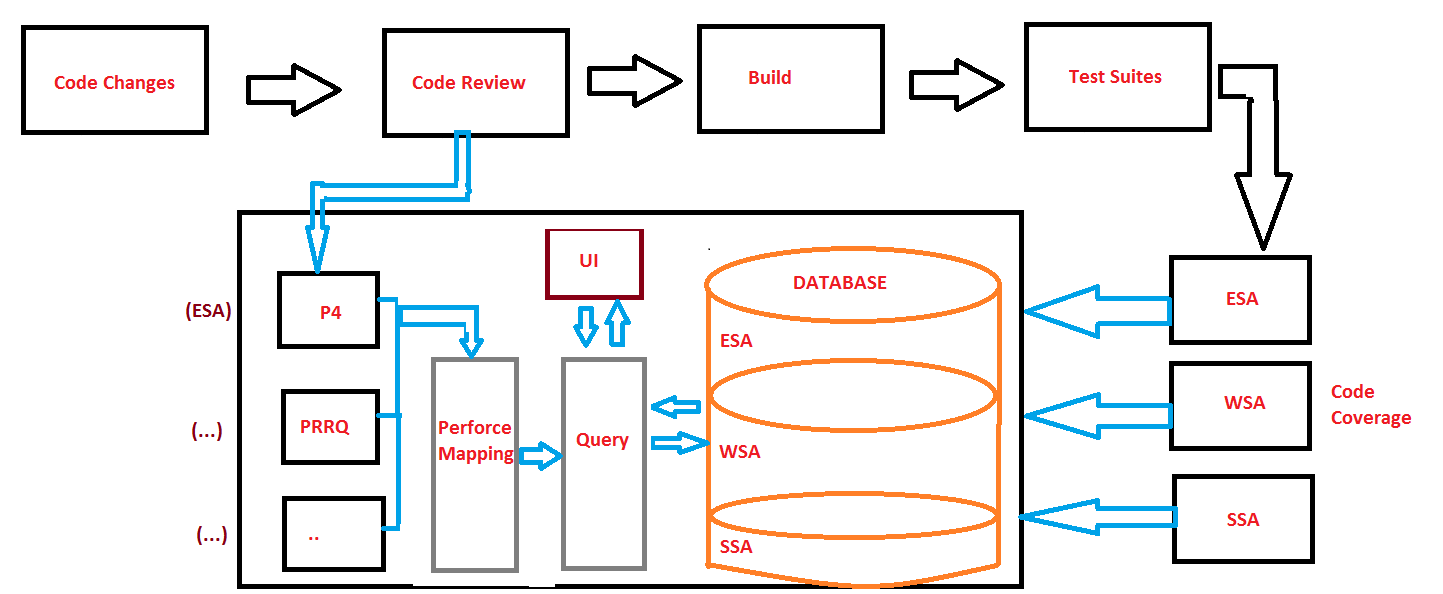
|  |  |  |
| --- | --- | --- |
| **Rohan Pandit**  **Technical Leader**  **Cisco Systems, Inc.**  **Bangalore** | **&** | **Chandrika Prasad**  **Assistant Professor**  **Department of Computer Science & Engineering** |

1. **Introduction**

Research studies show that software maintenance activity takes two thirds of the cost of software production. One of the expensive tasks, after maintenance is to establish confidence that changes made in the system are correct and has not developed any side effects in the already checked software. This assurance is achieved by performing regression testing. Regression testing is the re execution of test cases that have already been executed. In regression testing number of regression tests increases and it is impractical and inefficient to re-execute all test cases after every change made to software. Most of the times running an entire suite is not possible as it takes significant amount of time to run all tests in a test suite. Software testing and retesting occurs continuously during the software development life cycle to detect errors as early as possible, so various techniques have been proposed for minimizing test suite. Test Suite minimization techniques lower costs by reducing a test suite to a minimal subset that maintains equivalent coverage of original set with respect to particular test adequacy criterion

Regression testing calls for the execution of all the test cases tested before a change is made in the software. It takes a lot of time and resources and hence a technique is needed to prioritize the test cases so that only important test cases are re-executed thus saving the time and still not compromising with the quality of the software. Test case prioritization involves scheduling test cases so that performance of regression testing can be improved. To re execute all test cases after every modification in the code is an inefficient process. A technique is to execute the modified lines of code with minimum number of test cases. The proposed test case prioritization technique organizes the test case in a test suite in an ordering such that fewer lines of code need to be re executed thus faster code coverage is attained which would lead to early detection of faults.

The following figure gives an overview of the project designed overcome the problem mentioned above.



1. **Main Body**

The whole system which is being used in this project tries to take only those test suites and test cases and lines which are changed rather than debugging the whole system once a small change is done. Hence the project is called diff set based automation which means automation based on the line changes.

The system designed gives an overview of the project where different teams existing within the system has their own mechanism for building the test suites and test cases. But in general the project takes all the teams development changes and do the execution of required test suites. The test suites are collected by running the code coverage file and those are in turn converted to json using json converter which is shown in the above figure. The actual flow is in such way that the database keeps on refreshing once a week and changes made will be queried using the database where the database used here is the mongo DB database since it is NoSQL.

The following diagram gives an overview of the database (Mongo Db) used in this project.

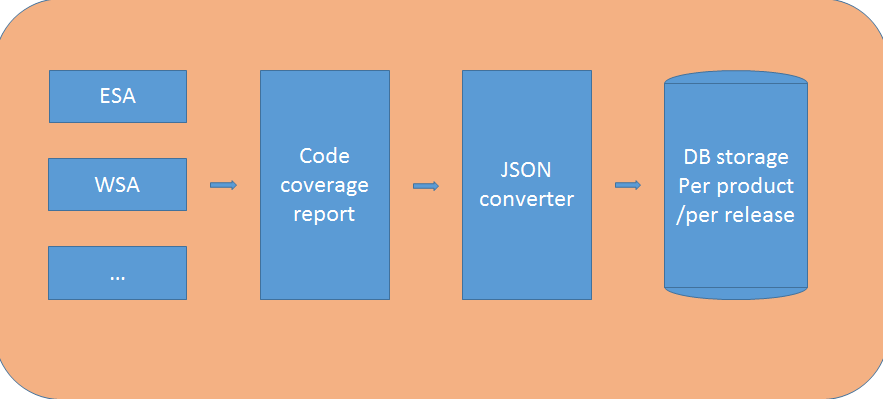


Figure 1: The overview of database storage

The following figure explains the clear picture of the flow of the Project and shows the complexity involved, both in the Automation system as well as in the Mapper.

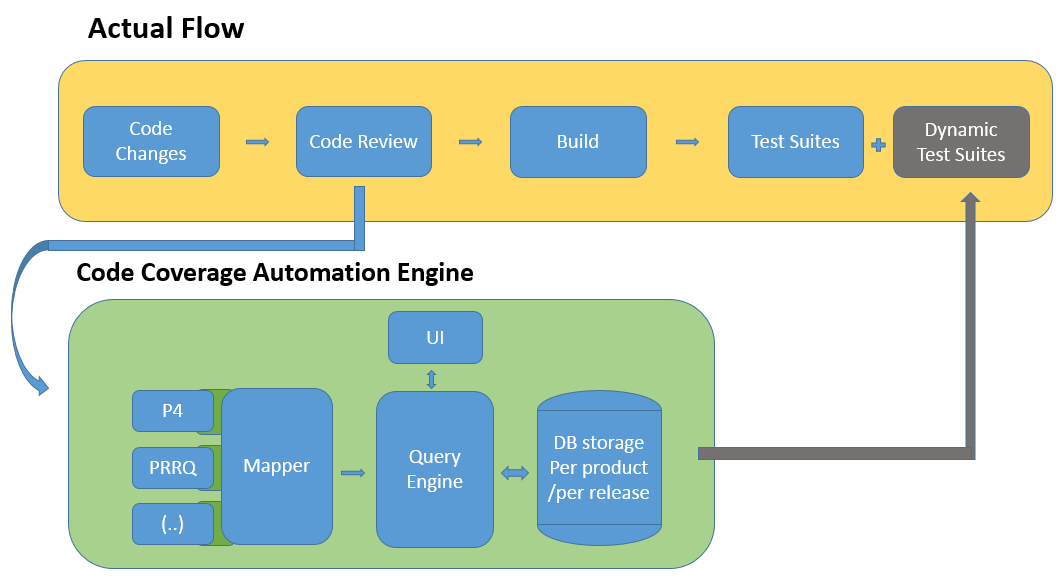


Figure 2: Overview of Cisco code coverage system and complexity

The code actual flow shown above consists of different parts and it is shown in a generic way which is applicable to all the teams in cisco. The following figure explains the flow for a specific team.

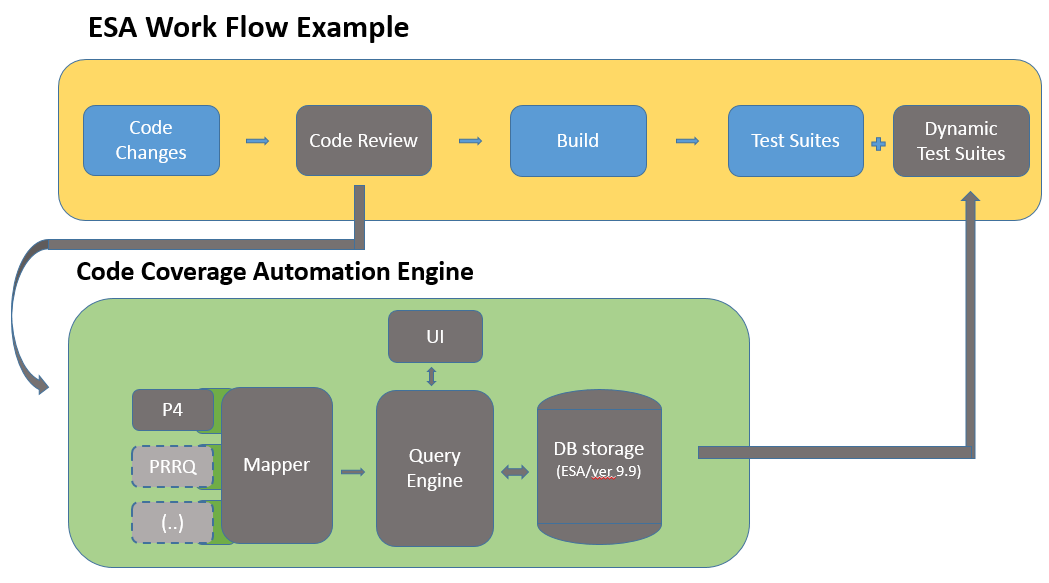


Figure 3: Overview of Cisco ESA Team code Flow system and complexity

1. **Conclusion**

This study highlights the details of the automated diff set system and brings out the benefits of automated identification of dependent test suites and test cases when compared to manual identification and running those manually which would be more difficult task. By implementing an effective automated diff set mechanism, the following business benefits can be achieved:

* Reduced downtime and costs associated with non-availability of systems and applications.
* Reduced labor costs.
* Improvement in Performance
* Effective use of available resources

1. **References**
2. <https://wiki.cisco.com/display/WSAENG/WSA+Test+Code+Coverage+Script>
3. <https://docs.mongodb.org/manual/?_ga=1.94425948.1319009703.1456767384>
4. <https://pypi.python.org/pypi/pexpect/>
5. <http://bglgrp0234-pod.cisco.com:8080/cisco/home.html>
6. <https://docs.python.org/2/library/xml.etree.elementtree.html>