

Internship At Intel Smart Regression to Reduce Regression Testing

Submitted By

Shubhra Singhal

18MCEC12



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INSTITUTE OF TECHNOLOGY
NIRMA UNIVERSITY

AHMEDABAD-382481

May 2020

Smart Regression to Reduce Regression Testing

Major Project

Submitted in partial fulfillment of the requirements
for the degree of
Master of Technology in Computer Science and Engineering

Submitted By

Shubhra Singhal

18MCEC12

Guided By

Dr. Madhuri Bhavsar



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INSTITUTE OF TECHNOLOGY
NIRMA UNIVERSITY

Ahmedabad-382481

May 2020

Certificate

This is to certify that the Major Project entitled “**Smart Regression to Reduce Regression Testing**” submitted by **Shubhra Singhal (18MCEC12)**, towards the partial fulfillment of the requirements for the degree of Master of Technology in **Computer Science & Engineering** of Nirma University, Ahmedabad is the record of work carried out by her under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this Major Project Part-II, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Dr Madhuri Bhavsar
Internal Guide, Professor & Head
CSE Department
Institute of Technology
Nirma University, Ahmedabad

Dr Priyanka Sharma
Professor & PG Coordinator(M.Tech - CSE)
CSE Department
Institute of Technology
Nirma University, Ahmedabad

Dr. R.N. Patel
Director (I/C),
Institute of Technology,
Nirma University, Ahmedabad

Certificate

This is to certify that the major project entitled “**Smart Regression To Reduce Regression Testing**” submitted by **Shubhra Singhal (18MCEC12)**, towards the partial fulfillment of the requirements for the award of degree of Master of Technology in **Computer Science & Engineering** of Nirma University, Ahmedabad, is the record of work carried out by her under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this Major Project , to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Mrs. Asha Balraj
Company Guide,
Infotech Manager,
Intel Corporation, Bangalore.

Statement of Originality

I, **Shubhra Singhal, 18MCEC12**, give undertaking that the Major Project entitled “**Smart Regression to Reduce Regression Testing**” submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in **Computer Science & Engineering** of Institute of Technology, Nirma University, Ahmedabad contains no material that has been awarded for any degree or diploma in any university or school in any territory to the best of my knowledge. It is the original work carried out by me and I give assurance that no attempt of plagiarism has been made. It contains no material that is previously published or written, except where reference has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

Signature of Student

Date:

Place:

Endorsed By:

Dr. Madhuri Bhavsar

Signature of Guide

Statement of Originality

I, **Shubhra Singhal, 18MCEC12**, give undertaking that the Major Project entitled “**Smart Regression to Reduce Regression Testing**” submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in **Computer Science & Engineering** of Institute of Technology, Nirma University, Ahmedabad contains no material that has been awarded for any degree or diploma in any university or school in any territory to the best of my knowledge. It is the original work carried out by me and I give assurance that no attempt of plagiarism has been made. It contains no material that is previously published or written, except where reference has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

Signature of Student

Date:

Place:

Endorsed By:

Mrs. Asha Balraj

Signature of Company Guide

Acknowledgements

It gives me immense pleasure in expressing thanks and profound gratitude to **Dr. Madhuri Bhavsar**, Professor & Head Of, Computer Science & Engineering Department, Institute of Technology, Nirma University, Ahmedabad for her kind support and providing basic infrastructure and healthy research environment.

It gives me an immense pleasure to thank **Mrs Asha Balraj**, who is a Infotech Manager and my Industry Guide and **Mrs. Sailaja Parthasarthy** who is my Manager, for their valuable guidance and continual encouragement throughout this work. The appreciation and continual support they had imparted has been a great motivation to me in reaching a higher goal. Their guidance has triggered and nourished my intellectual maturity that I will benefit from, for a long time to come.

A special thank you is expressed wholeheartedly to **Dr. R.N. Patel**, Hon'ble Director(I/C), Institute of Technology, Nirma University, Ahmedabad for the unmentionable motivation he has extended throughout course of this work.

I would also thank the Institution, all faculty members of Computer Engineering Department, Nirma University, Ahmedabad for their special attention and suggestions towards the project work.

-Shubhra Singhal

18MCEC12

Abstract

Once a product is developed then with the changing behaviour of the application as per the demand, it is must validate the changes made in a product. But the time needed to test those changes is considerably higher than the development time. As of now the nightly regression jobs run with a considerable amount of time for executing huge no of test cases and with new features added in time for that also new test cases required to keep product stable. It becomes vital to reduce the validation time and reduce overall resource allocation. This is where the smart regressions come where the total time is reduced by executing on-demand tests which are dependent on modified classes and contribute around 30-40% of the total number of tests written for that product. The faulty tests are detected and refactored again parallely along with the complete validation. This is achieved by various optimization techniques like by doing code coverage, the build deployment and report generation, in-order to achieve accurate reporting during the runtime of nightly scripts.

List Of Abbreviations

Wsn	Wireless Sensor Network
Jacoco	Java Code Coverage.
API	Application Programming Interface.
REST	REstful State Transfer .
VM	Virtual Machine.
RA	Retest All.
CSV	Comma Separated Values.
JSON	Javascript Object Notation.
RTS	Regression Test Selection.
XML	eXtensible Markup Language.
CICD	Continuous Integration and Continuous Delivery.

List of Figures

1.1	Regression Techniques	2
1.2	Stack Architecture	4
1.3	Example1 of Jacoco Code Coverage	8
1.4	Example2 of Jacoco Code Coverage	8
1.5	Jacoco Code Coverage Report	8
3.1	Collection Framework Hierarchy	16
3.2	Hash Set Example	17
3.3	Output of HashSet Example	18
3.4	Example of Index API using Map	19
3.5	Get API Example of getting document	19
3.6	Delete API example	20
3.7	Search API Example Using matchALLQuery	20
4.1	Architecture of The 1st Phase of project	25
4.2	Flow Diagram of 1st Phase	26
5.1	Different Devops Stages In Jenkins	29
5.2	Real World Case Of Jenkins	29
5.3	Basic Flow of Git	32
5.4	Basic Command Git 1	33
5.5	Basic Command Git 2	33
5.6	Basic Command Git 3	34
5.7	Basic Command Git 4	34
6.1	Working of Multi-threading	35
6.2	Block Diagram	37
6.3	Flow Diagram of 2nd Phase	37

List of Tables

1.1	Arguments passed to Generate Report	7
2.1	Analysis of Different Approaches	9
5.1	Reason of Jenkins Importance.	30

Contents

Certificate	iii
Certificate	iv
Statement of Originality	v
Statement of Originality	vi
Acknowledgements	vii
Abstract	viii
Abbreviations	ix
List of Figures	x
List of Tables	xi
1 Introduction	1
1.1 What is Regression Testing	1
1.1.1 How To Perform Regression Testing	1
1.1.2 Types of Regression Testing	2
1.1.3 Advantages	3
1.1.4 Disadvantages	3
1.2 What is Elasticsearch?	3
1.2.1 How Elasticsearch work?	3
1.2.2 Elasticsearch Stack	4
1.2.3 ELK Stack Architecture	4
1.2.4 Properties of Elastic search	4
1.2.5 Basic Terms Used in Elastic Search	5
1.2.6 Advantages	6
1.2.7 Disadvantages	6
1.3 Jacoco (Java Code Coverage Tool)	6
1.3.1 Features	6
1.3.2 Command Line Interface	7
1.4 Report Analysis	7
2 Literature Survey	9
2.1 Reason of Not Using these Methods	12

3	Technologies Used In Smart Regression Project	13
3.1	Java	13
3.2	Basic Concepts Of Java	13
3.2.1	Java Interfaces	13
3.2.2	Abstract Method and Classes	14
3.2.3	Collections in Java	14
3.2.4	HashSet With Example	16
3.2.5	Java Hashset Methods	18
3.3	ElasticSearch	18
3.3.1	Index API	19
3.3.2	Get API	19
3.3.3	Delete API	20
3.3.4	Search API	20
3.4	Junit	21
3.4.1	Benefits of Junit Testing	21
3.4.2	Junit Assertions	21
3.5	Perl Script	23
3.5.1	Functions and Statements	23
4	Getting Started on Work	24
4.1	Prerequisites	24
4.2	About Smart Regression Project	25
4.2.1	Explanation of Each Block	25
5	Tools Used In The Project	28
5.1	Jenkins	28
5.1.1	What is Jenkins	28
5.1.2	Advantages & Disadvantages of Jenkins	30
5.2	Git	31
5.2.1	Basic Commands	32
6	Implementation & Phase 2nd of Project Work	35
6.1	Multi-threading	35
6.2	Perl Script for Getting Exec File From Remote Server and Untar the latest Build	36
6.3	Description of second Phase	36
6.3.1	Explanation of Each Block in Flow Diagram	37
7	Verification and Validation of Software	39
7.1	Validation & Verification	39
7.2	Methods of Verification	40
7.2.1	Review	40
7.2.2	Walkthrough	40
7.2.3	Inspection	40
7.3	Methods of Validation	40
7.3.1	Testing	40
7.3.2	End Users	40
8	Conclusion	41

Chapter 1

Introduction

1.1 What is Regression Testing

Regression Testing is a testing technique to check whether after changes in code there should not have any adversely effects on existing features. This testing is to make certain that the product works fine with new capability, bug fixes or any changes in the current function. It is a software testing in which we execute test cases repeatedly to check the previous functionality will work fine after new changes has been made. Regression Testing is performed to verify after changes done or new feature is added that product is still working fine or not. Regression test runs on weekly releases when new Functional testing is added or some changes is done in old.

1.1.1 How To Perform Regression Testing

For software maintenance we perform error corrections, enhancement in features, optimizations and deletion of existing features. Because of these changes our system might not work correctly so we perform Regression testing to verify the correctness of system. There are various methods.

1. Retest All[5]

In this method we re-execute all the test cases that are written for the suite. This technique requires high amount of resources and time so it is an expensive technique.

2. Regression test Selection[5]

In this approach take a look at cases are classified in special components:

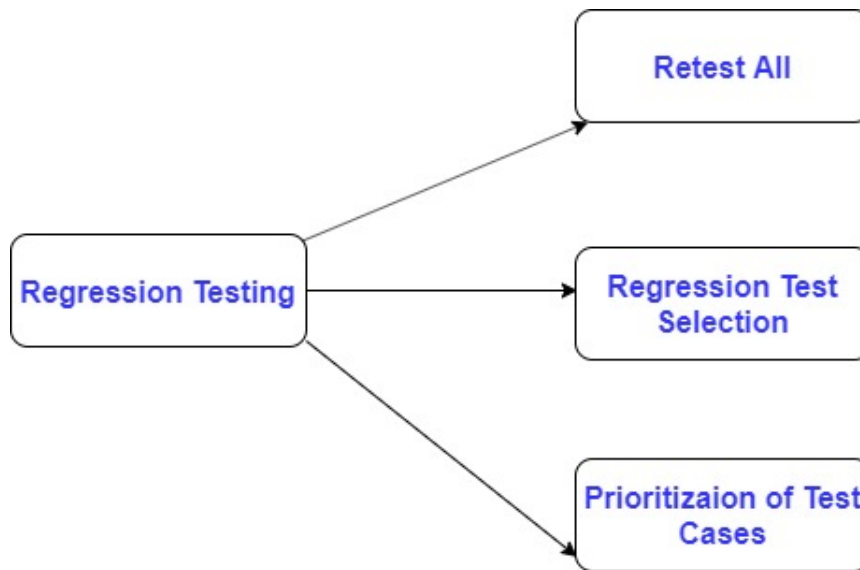


Figure 1.1: Regression Techniques

- Reusable Test Cases
- Obsolete Test cases

Reusable test cases are those which can be used in further regression cycle while obsolete can't be[3].

3. Prioritization of Test Cases[5]

In this method we divide test cases on the basis of their impact, critical & priority on software.

1.1.2 Types of Regression Testing

1. Unit Regression

Unit Regression is executed while Unit Testing is going on and code is tested in isolation, any dependencies are blocked in order that unit checking out can be achieved with none discrepancy.

2. Partial Regression

In this regression we affirm the code works high-quality when the unit is included with the already current code.

3. Complete Regression

Complete regression is carried out when wide variety of modules have been modified

and in existing characteristic we do adjustments. So to test the impact of changes within the whole code we do complete regression.

1.1.3 Advantages

- It improves the product quality.
- It ensures that after new changes or bug fixes the product is working fine.
- Automation tools can be used.
- It guarantees that the bug that are already detected and fix are not creatable now.
- We can't add new functionality when regression testing is on.
- It helps in performance optimization.

1.1.4 Disadvantages

- Without automation tool regression testing is not easy and will consume a lot of time.
- For very large test case suite it take a lot of time so the result of test will take so much time to come.
- Manual Regression testing is very tedious and lengthy process.
- If there is a small change in code then also we have to perform this.

1.2 What is Elasticsearch?

Elasticsearch is a distributed, open-supply search and analytics engine for each styles of information like textual, geospatial, dependent and unstructured. Elasticsearch is built on pinnacle of Apache Lucene and its first model came out in 2010. It provides easy Rest APIs, allotted in nature, proper velocity and scalability.

1.2.1 How Elasticsearch work?

Data in uncooked form is exceeded to Elasticsearch from a ramification of assets, net packages, machine metrics. In statistics ingestion raw records is parsed, normalized and enriched earlier than it is indexed in Elasticsearch. Once statistics is inserted, customers can run complicated queries and retrieve statistics.

1.2.2 Elasticsearch Stack

The ELK stack is a collection of 3 freely open products:

- **E** stands for “Elasticsearch”: whose purpose is used to store logs.
- **L** stands for “LogStash”: whose purpose is for processing and storing logs.
- **K** stands for “Kibana”: which works as a visualization tool, and allows us to write queries in easy way.

1.2.3 ELK Stack Architecture

- **Logs:** Server logs which are required to be analyze and discover[8].
- **Logstash:**Collect logs, statistics and events. It parses and transforms information [8].
- **ElasticSearch:** The transformed information from Logstash is Store, search and indexed[8].
- **Kibana:** It helps to explore, visualize and share the information[8].

When handling very massive quantity of statistics we'd need Kafka, RabbitMQ for buffering and resilience. For security, nginx can be used.

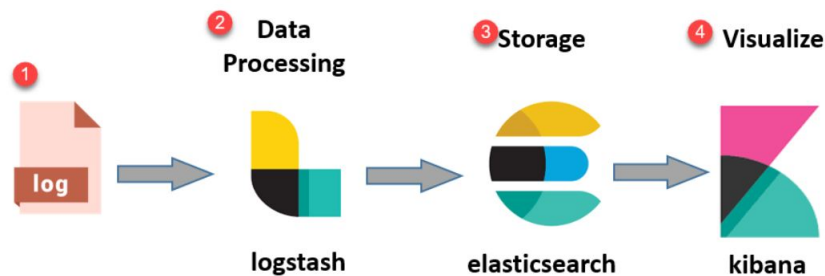


Figure 1.2: Stack Architecture

1.2.4 Properties of Elastic search

- It is open source search server which is written using Java.
- Full-text search is possible.
- Real-time search possible.

- It is schema-free, has REST API web interface.
- It guarantees availability by providing replicated searchable.
- Data is stored in JSON format.
- Supports multi-language & Geolocation support.

1.2.5 Basic Terms Used in Elastic Search

1. Node

Node is an instance of elasticsearch, it is created when we start elasticsearch.

2. Cluster

The collection of nodes make cluster in which we keep data and help to provide joined indexing and search capabilities.

3. Index

Index is a collection of documents, it is like database in MySQL. It is very useful to perform the operations like data processing, searching, updating and deleting. We can define more than one index in one cluster.

4. Document

It is basic unit of information like row in MySQL. It is stored in JSON format (key,value) pair. Every document in elasticsearch is associated with unique ID and a type.

5. Shard

Every index is split into several shards so that data can be distributed. Shards is an independent node, which can be store on any node. Primary shard is the horizontal part of an index and we create replicas of these shards[4].

6. Replicas

For availability and good performance elasticsearch allows to create replicas of their indexes and shards. When one shard is down data searching, update or deletion can be performed on replicas.

1.2.6 Advantages

- As Elasticsearch is developed on Java, so it can be run on most of the platforms.
- It works on real time, it means when document is added after one second it is searchable.
- It is distributed, and easily scalable and can be integrate with any big organization.
- Creating backup is easy.
- It uses JSON objects as responses so we can use this server with different programming languages.

1.2.7 Disadvantages

- It doesn't support mutli-language like CSV, XML formats.
- Very rarely, there is a chance of split-brain problem.

1.3 Jacoco (Java Code Coverage Tool)

Jacoco is a “JAVa COde COverage tool” it is used to measure the no of lines, branches and instructions is covered by the test case. There are unique code coverage tools avaiable for Java. The Jacoco is a freely to be had code coverage library for Java, which has been created by using EclEmma. Previously Jacoco works with only JDK but now we can use this for OpenJDK also.

1.3.1 Features

- It provide coverage analysis for instructions, branches, lines, methods & types.
- It is based on Java byte code so it can work without source files.
- We integrate Java agent primarily based on-the fly instrumentation.
- It can produce report in different formats like CSV, HTML & XML.
- Support different JVM languages.
- We can integrate this with gradle-plugin to collect coverage information and create reports.

1.3.2 Command Line Interface

To perform basic operations using command line the Jacoco provides *Jacoco Command Line Interface* (jacococli). All the dependencies are included in jacococli.jar file.

To get **report** the syntax is:

```
java -jar jacococli.jar report [execfiles ...] -classfiles path [-csv file] [-encoding charset] [-help] [-html dir] [-name name] [-quiet] [-sourcefiles path] [-tabwith n] [-xml file] [2]
```

We can generate reports in different formats from exec and Java class files.

Option	Description
execfiles	list of JaCoCo *.exec files to read
-classfiles	location of Java class files.
-csv	output file for the CSV report
-encoding	source file encoding (by default platform encoding is used)
-help	show help
-html	output directory for the HTML report
-name	name used for this report
-quiet	suppress all output on stdout
-sourcefiles	location of the source files
-tabwith	tab stop width for the source pages (default 4)
-xml	output file for the XML report

Table 1.1: Arguments passed to Generate Report

1.4 Report Analysis

My report shows that 78% coverage and red mark shows the untouched part. Jacoco reports help us to analyze the code coverage by using diamonds with colors for branches and background color for lines. This is shown in fig 1.4

- **Red Diamond** means that no branches are covered by test.
- **Yellow Diamond** means that the code is partially covered some part is covered some is not.
- **Green Diamond** means all branches are covered during this test phase.

The same color code applied for the the background but for line coverage it provides different type of metrics:

Calculator.java

```

1. package com.demo;
2.
3. public class Calculator {
4.     public static int addNumber(int a,int b){
5.         return a+b;
6.     }
7.     public static int subtarctNumber(int a, int b){ return a-b;}
8.
9.     public static int multiplyNumber(int a,int b) { return a*b;}
10.    public static int divideNumber(int a,int b){return a/b;}
11. }

```

Figure 1.3: Example1 of Jacoco Code Coverage

Palindrome.java

```

1. package com.demo;
2.
3. public class Palindrome {
4.     public boolean isPalindrome(String inputString) {
5.         if (inputString.length() == 0) {
6.             return true;
7.         } else {
8.             char firstChar = inputString.charAt(0);
9.             char lastChar = inputString.charAt(inputString.length() - 1);
10.            String mid = inputString.substring(1, inputString.length() - 1);
11.            return (firstChar == lastChar) && isPalindrome(mid);
12.        }
13.    }
14. }

```

Figure 1.4: Example2 of Jacoco Code Coverage

Calculator






Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
subtarctNumber(int, int)		0%		n/a	1	1	1	1	1	1
addNumber(int, int)		100%		n/a	0	1	0	1	0	1
multiplyNumber(int, int)		100%		n/a	0	1	0	1	0	1
divideNumber(int, int)		100%		n/a	0	1	0	1	0	1
Calculator()		100%		n/a	0	1	0	1	0	1
Total	4 of 19	78%	0 of 0	n/a	1	5	1	5	1	5

Figure 1.5: Jacoco Code Coverage Report

- **Lines Coverage** Indicates the amount of code that has been protected on the premise of number of Java byte code instructions.
- **Branches Coverage** shows the code coverage on the basis of branches like if/else and switch statements.
- **Cyclomatic Complexity** Suggests the complexity on the basis of wide variety of paths required to cover all of the possible paths.

Chapter 2

Literature Survey

Table 2.1: Analysis of Different Approaches

Sr. No	Author and Year	Title	Methodology	Conclusion or Future Work
1.	Ansari, Ahlam S.A.,Devadkar Kailas K.,Gharpure Prachi 2013	Optimization of test suite-test case in regression test[6]	Test Suite Refinement Technique is cross-combination of Test Suite Minimization based on Specification and one more is Test Suite Prioritization based on Risk Exposure	It is effective and gain considerable check suite test case length discount. Reduce cost and time.Lessen maintenance effort Ensure reliability by giving plentiful no of test cases.

2.	Jahanvi & A.Singh-2014	Efficient Regression Test Selection & Recommendation Approach for Component Based Software [10]	Approach- Regression test Selection and Recommendation.Using UML state chart and sequence diagrams.Case Study On- ATM	RTSR selects decrease variety of check instances than authentic test suite. The percentage reduction of 61.9 % is completed inside the regression test suite. Future Work - Apply procedure on large systems
3.	S.K. Mohapatra & M Pradhan-2015	Finding Representative Test Suit for Test Case Reduction in Regression Testing [13]	Use Genetic Algorithm to reduce no of test cases. Create a minimal set of test cases to satisfy all the code coverage requirements.	The generated take a look at suite is minimized greatly. Reduce cost of regression checking out. Improve performance of check suite.

4.	C.P.Indumathi & S.Madhumati -2018	Cost Aware Test Suite Reduction Algorithm for Regression Testing[9]	Use Genetic Algorithm to prioritize the test case Maximum Frequent test set(MFTS) used to find maximum no of test suites. Average percentage of fault detection(APFD) metric is used to increase the test suite rate of fault detection	In this present work they locate the effectiveness of genetic prioritization for regression trying out.The obtained check collection is much less expensive and much less critical then other.Lessen upkeep effort The proposed MFTS algorithm for take a look at suite discount also addressed the test metrics: size, requirement coverage, fault detection and execution time.
----	-----------------------------------	---	---	---

5.	D.Marijan, & M.Liaaen-2018	Practical selective regression testing with effective redundancy in interleaved tests	Combine test coverage metrics and execution history for reducing regression in (High Computing Software) HCS in CI Testing done on large scale video confrencing software [12]. On three basis experiments are done: First two done on the basis of fault detection effectiveness. Remove totally redundant and partially redundant in first approach. Compare with retest all. In 3rd compare using random test selection.	In approach 1and approach 2 the time effectiveness is good without compromising fault effectiveness.Higher reduction is achieved for higher degree of feature interaction test suites.In approach 3 the fault-detection effectiveness of a test suite is improved compare to random test.
----	----------------------------	---	---	---

2.1 Reason of Not Using these Methods

In all these papers basically the researchers is trying to reduce no of test cases because of there repetition or trying to reduce no of test case by prioritization or obsolete the test case.

But with my idea of doing this project when we don't have repeated test case and all test need to be run, because all test are important and testing the feature in different environment. So instead of reducing test case we are using the different approach to reduce the regression testing.

Chapter 3

Technologies Used In Smart Regression Project

To develop the code for Smart Regression Project I need to learn few technologies so that I can use them to implement the project.

3.1 Java

We use the Java language to develop my project. Java is high level language and its platform independent. It is object oriented, simple and secure to use. Java is developed by thinking the concept of Write Once Run Anywhere, which runs on VM [11].

3.2 Basic Concepts Of Java

Many concepts of Java I learn to use in my project there are some that I am explaining in brief:-

3.2.1 Java Interfaces

Interface is like a Java class and defined via keyword *interface*. Interface allow us to use multiple inheritance. All methods in interface are implicitly public and abstract. An interface can have many methods and all are abstract. We can't extend an interface it is implemented by class using keyword *implements*. No constructors are defined in interface.

Example

```
interface Person {  
    public void talk();  
    public void eat();  
    public void drink();  
}
```

3.2.2 Abstract Method and Classes

Abstraction is one of the OOPS concept, abstraction is use to hide the features and only showing the essential features. In Java , an abstract class is declared by using keyword *abstract*. Abstract method have no method body. In abstract class we can have abstract and non-abstract methods. An abstract class can have constructors and static method also.

Example

```
abstract class GraphicsObj {  
    int length, breadth;  
    void moveTo(int length, int breadth) {  
        .....  
    }  
    abstract void draw();  
    abstract void calculate();  
}
```

3.2.3 Collections in Java

Collection is a set of discrete objects and defined as single unit. Two packages **java.util.Collection** and **java.util.Map** are two main collection and map interface that we use. The collection hierarchy framework is shown in fig 3.1

Collection Framework Hierarchy

1. Iterable interface

It is the root of collection classes. The collection interface extends Iterable and therefore all other implement this Iterable interface. One abstract method i.e.,
Iterator <T >iterator()

2. **Collection Interface**

Collection interface is implemented by all the classes in collection framework.

3. **List Interface**

It is child interface of collection. In this we can store list type data structure. It can consist of duplicate values. There are four classes that implement List:

- ArrayList
- LinkedList
- Stack
- Vector

4. **Queue**

It uses the concept of first-in-first-out order. It can be defined as ordered list which is used to hold elements. There are various classes like:

- **PriorityQueue Interface**

It process the elements according to their priority.

- **Deque Interface**

It extends Queue interface. In this we can remove and add elements from the both side. Deque stands for Double ended queue.

- **ArrayDeque**

This is also like Deque, it is faster than arraylist and stack.

5. **Set Interface**

It represents the unordered set of elements in which we can't store duplicate elements. We can store at most only one null value. Set is implemented by the three classes:

- **HashSet**

Hashing method is used to store in HashSet.

- **LinkedHashSet**

It also contain unique elements and maintains the insertion order and allow null elements.

- **SortedSet Interface**

In this elements are arranged in increasing order.

- **TreeSet**

It uses a tree for storage and implements Set interface. The access and retrieval time is fast and data is stored in ascending order.

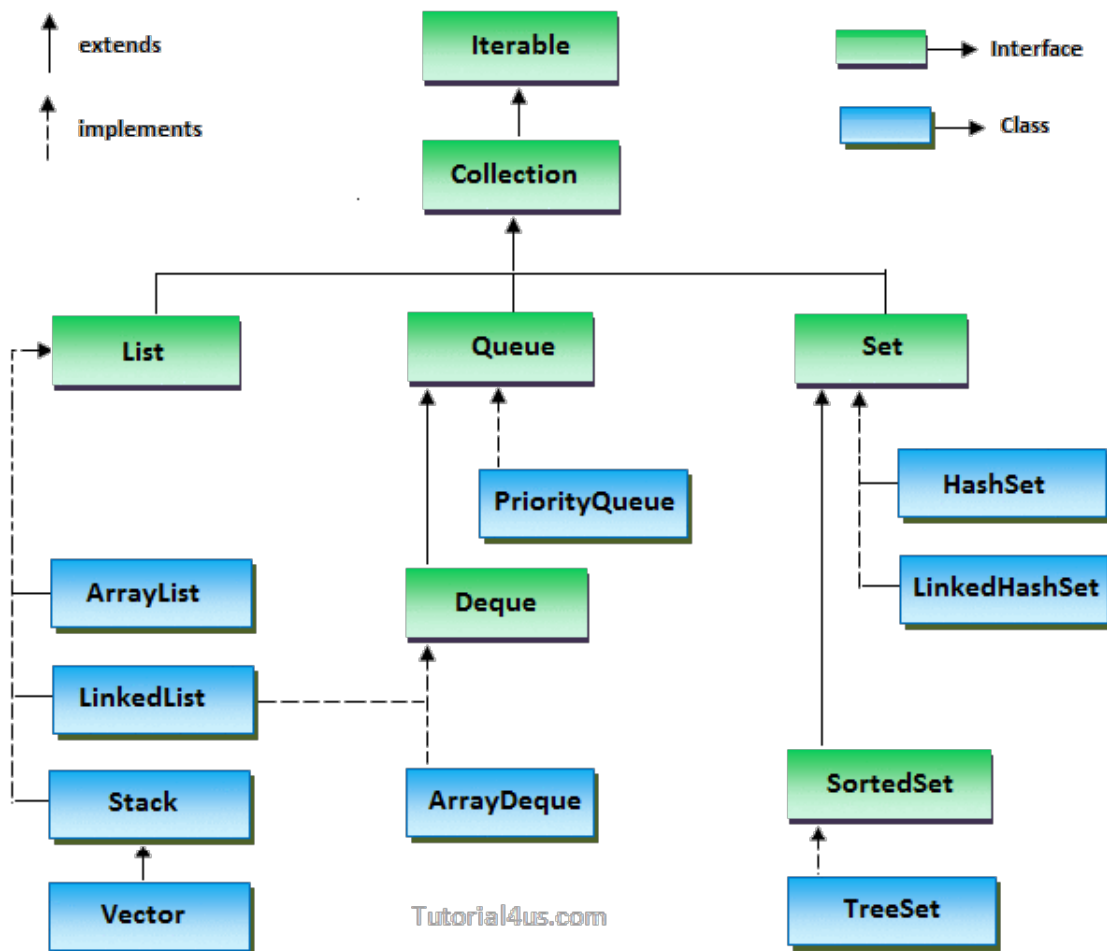


Figure 3.1: Collection Framework Hierarchy

3.2.4 HashSet With Example

In our project we used HashSet to remove the duplicate values. The brief Description of its working is explained here:

- HashSet class extends AbstractSet and implements Set, Cloneable and Serializable interfaces[7].
- Hash Set is an unordered collection.

```

package collections.framework;

import java.util.HashSet;
import java.util.Set;

public class HashSetDemo {
    public static void main(String[] args) {
        // creating a HashSet
        Set<String> citySet = new HashSet<String>();
        // Adding elements
        citySet.add("London");
        citySet.add(null);
        citySet.add("Tokyo");
        citySet.add("New Delhi");
        citySet.add("Beijing");
        citySet.add("Nairobi");
        citySet.add("New Delhi");
        citySet.add(null);
        // iterating HashSet
        for(String city : citySet){
            System.out.println("City- " + city);
        }
    }
}

```

Figure 3.2: Hash Set Example

- It calculate the hash value and decide where the element to be stored in HashSet.
- Only one null value is allowed.
- It is not synchronized so if need to use with multi-threaded so we need to synchronized it externally.
- The iterators back by way of HashSet's iterator method are fail-speedy: if the set is modified at any time after the iterator is created, in any way except through the iterator's own remove technique, the Iterator throws a ConcurrentModificationException.

In fig 3.3 is shown the output from where we can verify that:

- Insertion order can not be maintained inside the hashSet.
- Duplicate values also not allowed, as shown 3.2 New Delhi though inserted twice but it is added only once.

- Null element also can be added only once.

```
"C:\Program Files\Java\jdk1.8.0_212\bin\java.exe" ...
City- null
City- Beijing
City- New Delhi
City- Nairobi
City- Tokyo
City- London
Picked up _JAVA_OPTIONS: -Djava.net.preferIPv4Stack=true
```

Figure 3.3: Output of HashSet Example

In fig 3.2 is shown the hash set example. All the collection framework are common place so we are able to specify the sort of elements the HashSet goes to keep at the introduction time itself. HashSet created here is supposed to store String type data only.

3.2.5 Java Hashset Methods

- **add (E e)**- Adds the defined element to this set if it isn't so far present.
- **clear()**- Removes each and every single elements from this set.
- **contains(Object o)**- Returns true if only the set has contains the specified element.
- **isEmpty()**- Returns true if only set has not contains any elements.
- **iterator()**- Returns an iterator over all the elements in the set.
- **remove(Object o)**- It will remove the specific element from this set if it is present.
- **size()**- It will return the total number of elements in the set (its cardinality).
- **spliterator()**- Creates a late-binding and fail-fast Spliterator over all the elements in the set.

3.3 ElasticSearch

The overview of elasticsearch is explained in Chapter 1 To using Elastic search with Java we had use the Elasticsearch Rest APIs. There are some APIs that I used frequently and explaining here.

3.3.1 Index API

The index API lets in one to index a typed JSON report into a particular index and make it searchable.

Generate JSON Document There are different approaches for generating the JSON document.

1. Manually by using the usage of native byte[] or as as String.
2. We can Use a *Map* that will spontaneously convert this into JSON format.
3. With help of third party library like *Jackson* we can to serialize the beans.
4. Using existing built in helpers like `XContentFactory.jsonBuilder()`.

Example Using Map

Here I am giving example of Using Map as using this method we defined the mappings in the project. The get API allows to get a typed JSON document from the index based

```
Map<String, Object> json = new HashMap<String, Object>();  
json.put("user", "kimchy");  
json.put("postDate", new Date());  
json.put("message", "trying out Elasticsearch");
```

Figure 3.4: Example of Index API using Map

on its id.

3.3.2 Get API

The Get API allows to get a typed JSON document from the index based on its id.

The following example gets a JSON record from an index known as twitter, underneath a type referred to as `_doc`, with identification valued 1:

```
GetResponse response = client.prepareGet("twitter", "_doc", "1").get();
```

Figure 3.5: Get API Example of getting document


```
DeleteResponse response = client.prepareDelete("twitter", "_doc", "1").get();
```

Figure 3.6: Delete API example

3.3.3 Delete API

The delete API allows one to delete a typed JSON document from a specific index based on its id.

The example shown in fig 3.6 will delete the JSON document from an index called as twitter, from a type called _doc, with identification valued 1:

3.3.4 Search API

The search API allows one to execute a search question and returned search hits that match the query. It can be carried out across one or more indices and throughout one or extra types. The query may be supplied the usage of the query Java API. The body of the quest request is constructed using the SearchSourceBuilder.

This is the most basic example of add a query to the request:

```
SearchRequest searchRequest = new SearchRequest(); ❶  
SearchSourceBuilder searchSourceBuilder = new SearchSourceBuilder(); ❷  
searchSourceBuilder.query(QueryBuilders.matchAllQuery()); ❸  
searchRequest.source(searchSourceBuilder); ❹
```

Figure 3.7: Search API Example Using matchALLQuery

1. Create the *SearchRequest* without arguments this will check for all indices.
2. Most search parameters are added to the *SearchSourceBuilder*. It offers setters for everything that goes into the search request body.
3. Add a *match_all* query to the SearchSourceBuilder.
4. Add the SearchSourceBuilder to the SearchRequest.

3.4 Junit

JUnit is a unit checking out framework for the Java programming language. JUnit has been critical in the development of check-pushed development, and is one of a circle of relatives of unit testing frameworks that is collectively known as xUnit that originated with SUnit.

Dependency of Junit in Gradle

```
dependencies {  
testCompile group: 'junit', name: 'junit', version: '4.12'  
}
```

3.4.1 Benefits of Junit Testing

- It confirm that small unit of code work correctly.
- That small unit can be a java class, method on a java class or even a static code.
- Sometimes with one target code it tries to several classes once. Those type of unit tests are often known as component test.
- The term expected and actual will be used more here.
- Done manually before test libraries appeared.
- Provide quick feedback.
- Automated regression checking - We can run all the test before we deploy the system.
- The point of writing automated tests not so much to verify that the code works now.
- The point is to verify on an ongoing basis that the code continues to work in the future.

3.4.2 Junit Assertions

Assert is a method useful in determining Pass or Fail status of a test case, The assert methods are provided by the class **org.junit.Assert** which extends **java.lang.Object**

class.

Junit Assert Methods

1. **Boolean** If you want to test the Boolean conditions, you could use the following assert methods:-

- `assert True(condition)`
- `assert False(condition)`

2. **Null Object** If you wish to verify the initial value of an object or a variable, you can use the following methods:

- `assert Null(object)`
- `assert NotNull(object)`

3. **Identical** If you wish to verify that the objects are identical or different, use these methods:

- `assert Same(expected, actual)` - It will return true if `expected == actual`
- `assert NotSame(expected, actual)`

4. **Assert Equals** To check the equality between two of the objects we can use the following methods:

- `assert Equals(expected, actual)`- It will return true if: `expected.equals (actual)` returns true.
- `assert ArrayEquals(expected, actual)`- If you desire to test equality of arrays.
- `assert ArrayEquals(expected[i],actual[i])`- This method should be used iff arrays have the same length, for each valid value for i.

5. **Fail Message**- If you wish to throw any type of assertion error, you have this `fail()` that always results in a fail.

`fail(message);`

3.5 Perl Script

Perl is a scripting programming language. We use Perl for writing regression script because it is stable and cross-platform programming language.

Perl Features

- Perl adapts the best features from many other languages, like as C, awk, sed, sh, and BASIC, among different others.
- Perl's Database Integration Interface (DBI) allows third-party databases like Oracle, Sybase, Postgres, MySQL and others.
- Perl can work with other scripting language like HTML, XML, and other mark-up languages.
- Perl promotes both procedural as well as object-oriented programming.
- Perl at a time is the most popular web programming language because of its text manipulation feature and fast development cycle.

3.5.1 Functions and Statements

Here I am explaining some of those functions and statements that I used in my project.

use feature qw (say)

This is used to print the statement but the difference between print and say is, say will enter the new line everytime when we use this method 'say'.

system(\$(command))

This statement is used to run the linux command through the perl script. Example of running linux command is:

```
my $cmd = 'rm -rf filename.txt';  
system($cmd);
```

Chapter 4

Getting Started on Work

4.1 Prerequisites

I have joined Intel as Graduate Technical Intern in HPC Middleware Validation team. The team is responsible for validating products like storage management, distributed computing tool etc. I am working on distributed computing tool, firstly they had given me KT on this distributed computing product, how it works, how to submit jobs in this environment and manage those jobs. How internally this product work all those training are maintained below:

- Setting up the Linux machine.
- Download and install the product.
- Configuring the product for running the task.
- Team uses different tools like git, Jenkins. Git is a version control system. Jenkins is used to run regression scripts.
- Configure jobs on different queues.
- Create queues with different priority and different resource requirement like OS and memory and then schedule jobs according to their requirement.
- Installing IntelliJ IDE and setting up the proxies.

After around 1 month I started working on Smart Regression Project, for this project I have gone through some trainings which are:

- Basic Java
- Junit
- Java Design patterns
- Learning Perl script
- Elasticsearch engine database

4.2 About Smart Regression Project

Architecture of Project The architecture of the project is shown in fig 4.1. Here the n is the no of test cases. As if there are let say 10k test so there will be 10k exec files and XML files correspond to each exec file. This project is divided into two phases. In

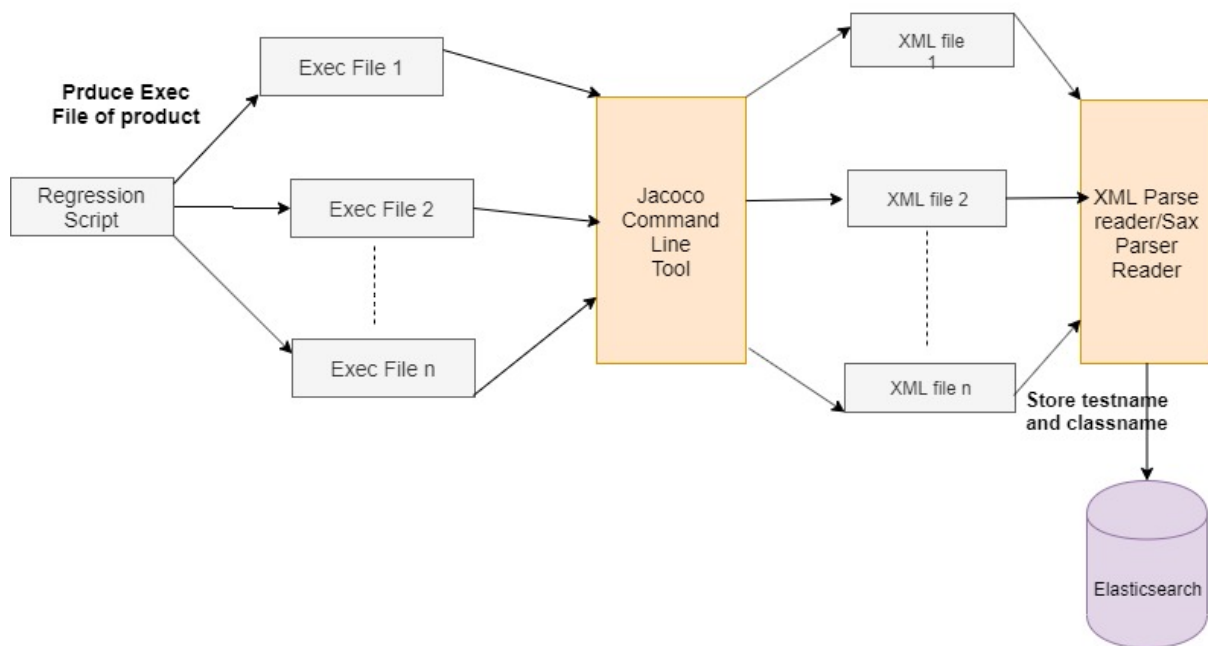


Figure 4.1: Architecture of The 1st Phase of project

the first phase the data is populated in Elasticsearch. In second phase we retrieve the test-name for the newly committed class.

4.2.1 Explanation of Each Block

The flow of the project is shown in fig 5.1. The detailed explanation of each block is given here:

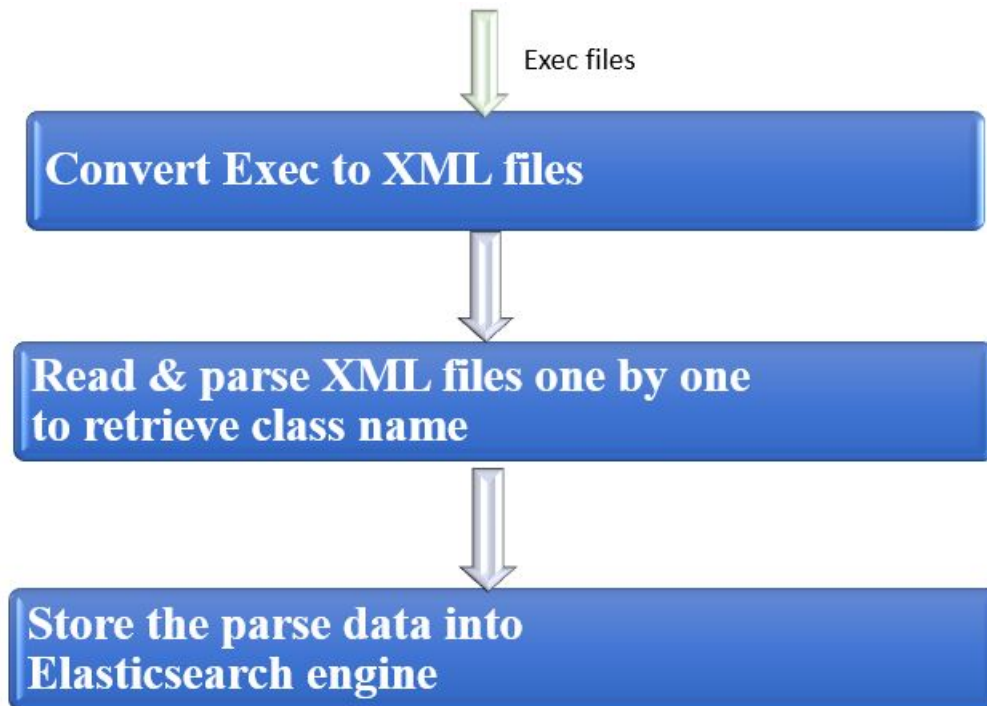


Figure 4.2: Flow Diagram of 1st Phase

1. **Generating Exec Files**

The Exec Files will be generated using Jacoco, we integrate the Jacoco with the product. To use the Jacoco, I integrated the jacoco with the source code so, when we run the test it will produce the .exec file of the code.

For creating exec files of each test case, we integrate the Jacoco command with the source code which will create exec file of each and every test case whenever the regression test is run. While we run the test it will set Jacoco Agent in motion, to create the code coverage report in *binary format*, the report will be stored in the target location - target/jacoco.exec. The name of Exec will be same as name of test case.

2. **Converting exec to xml file**

For converting the Exec file to Xml format I use the Jacoco command line tool, which will take all the Exec files and convert them into the XML file. The syntax of command that is use to convert the file in xml format is:

```
java -jar NFS_PATH/jacoco/lib/jacococli.jar report  
NFS_PATH/ExecFiles/filename.exec -classfiles NFS_PATH/product.jar -xml  
NFS_PATH/xmlfiles/filename.xml
```

3. Read and Parse XML files one by one to retrieve Class name

The converted XML files need to parse to get the name of classes. The classes that are touched by the test are stored in the xml files. The coverage value is checked by branch, instructions and lines. The covered ratio can be seen by those parameters. By parsing the xml files we retrieve the name of classes.

The data coverage report has information on the basis of lines, branch, instruction so the name of class is repeated so to remove the duplicate values we use the Set which is a class of collection framework.

4. Store Data In Elasticsearch

The class name which is parsed from the XML are store in Hashset and then the name of test is mapped with class and stored in Elasticsearch. To insert the data in elasticsearch first the connectivity with the Elasticsearch is established then the data is stored in it by using Elastic Search APIs. The APIs that I used in this project are all described in chapter 3. As when new build comes there is a chance new test case come so to update the test case we use the Search API and if data is not present in Elasticsearch then only we insert the data.

Chapter 5

Tools Used In The Project

5.1 Jenkins

5.1.1 What is Jenkins

Jenkins is a CI/CD tool. It is an open source and freely available server. It is an automation tool which is written in Java Language. It helps to automate the parts of software development like building, testing and deploying, and facilitating the continuous integration and continuous delivery[14]. Jenkins achieves Continuous Integration with the assist of plugins. Plugins allows the integration of Various DevOps stages. If you want to integrate a particular tool, you want to install the plugins for that device. For example: Git, Maven 2 project, Amazon EC2, HTML publisher and so forth. The figure in 5.1 is showing the different devops stages of Jenkins.

Continuous Integration and Continuous Delivery

Continuous Integration afterwards a code devotes, the software is constructed and tested proper away. In a big task with many developers, commits are completed so commonly in a day. After each dedicate the code is built and tested. If the take a look at is passed, then construct is tested for deployment. If the deployment is a success, the code is driven to manufacturing. This devotes, build, test, and installation is a continuous procedure and for this reason the call is continuous integration continuous deployment. The real world example of jenkins is shown in fig 5.2

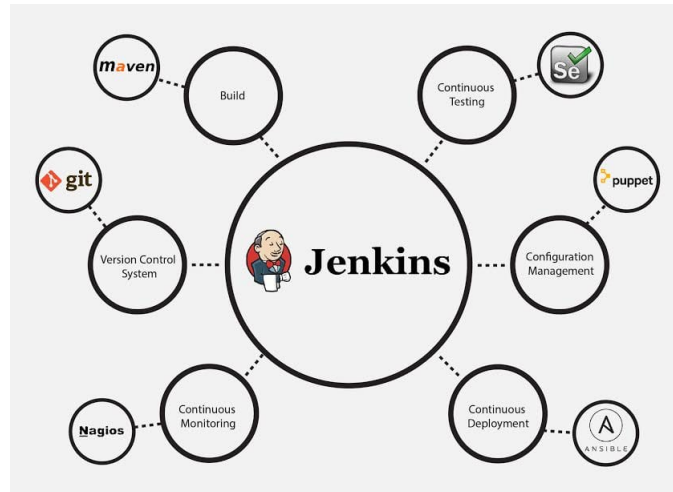


Figure 5.1: Different Devops Stages In Jenkins

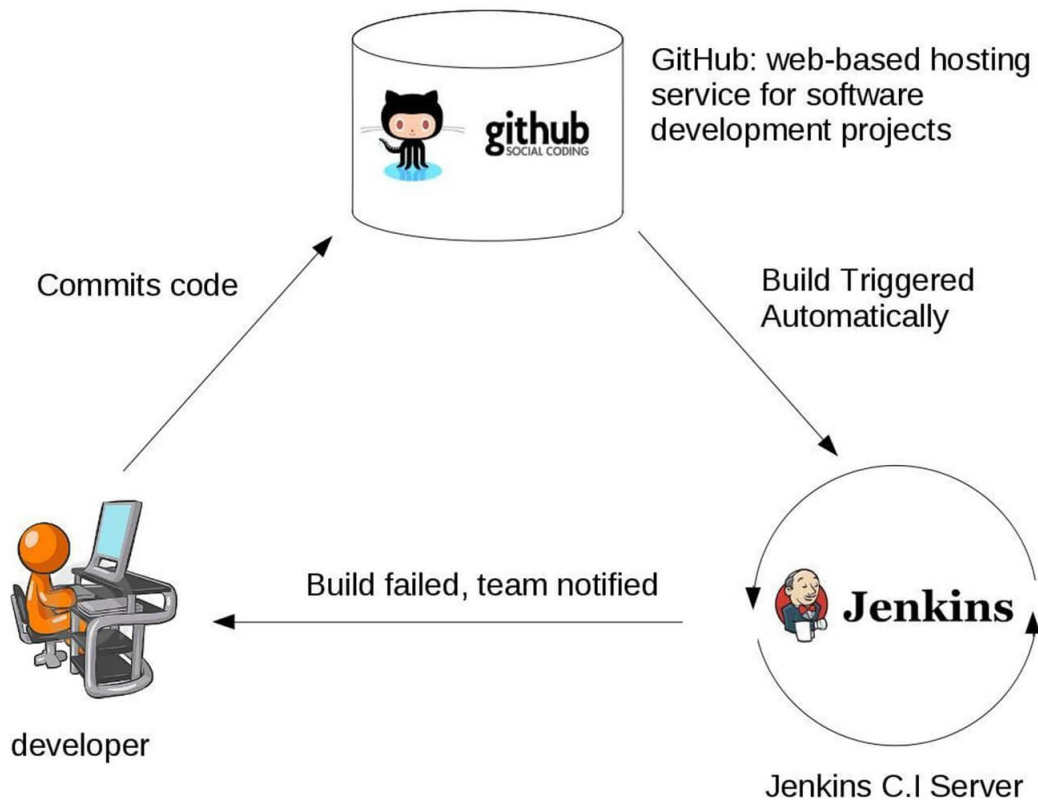


Figure 5.2: Real World Case Of Jenkins

Why Jenkins is Important

To understand this concept that why Jenkins tool is that much important lets take an example of an industry where some product is going to develop by some 10 developers in which 7 take 15 days and 3 take 20 days to complete the project then with this table shown in 5.1 see the difference before Jenkins and after Jenkins.

Before Jenkins	After Jenkins
When all Developers will complet their assigned coding tasks, then they use to commit their code once all at the same time. Later,only Build is tested and deployed.	Now the code can be built and tested as soon as Developer commits code.It will not for all commits by each and every developer Jenkins will build and test code many times during the day.
As the code built all at same time, some developers have to wait until all other developers finish coding to check about build.	The code can be built immediately after any of the Developer will commits.
It is not easy to isolate the code, and then find the defects and fix it.	Using Jenkins the build is deployed after each commit of any developer so find the defect is easy and as so to fix that bug.
Development phase is slow.	Development phase is fast and it is easily adaptable to add new features when required.

Table 5.1: Reason of Jenkins Importance.

5.1.2 Advantages & Disadvantages of Jenkins

Advantages

- It is an open source tool.
- It has more than 1000 plugins which ease your work.
- It provides great network and community support.
- It promotes cloud based architecture also so we have facility to deploy Jenkins in cloud-based platforms also.
- It is popular mainly because it is developed by developers for developers easiness.

Disadvantages

- The interface is not latest one and not that much user friendly.
- To learn each and every feature in detail it take time.

- Installing and configuring Jenkins is also one of the tough part so that's the some people don't prefer that.
- Jenkins is useful and powerful tool but to managing and handling the Jenkins server we need a special attention and some times require dedicated developer.

5.2 Git

Version control system allows developers to work simultaneously and keep history of every version. There are basically 2 types of Version Control System:

- Distributed Version Control System
- Centralized Version Control System

Git is a free and open supply disbursed model control device designed to address the whole thing from small to very massive initiatives with pace and efficiency[1].

Advantages

- Free and Open Source
- Fast and Secure
- Easily we can create branches
- Easy to learn and understand
- No specific powerful hardware is needed

Git Basic Terminologies

1. Working Directory

Working directory is the place where we check out the files. Do modification and add the files which we required.

2. Staging Area

It is the area between local and remote repository. The files which are committed all present in the staging area. Only the files which are committed can be move to remote location. All modified files will remain in working directory.

3. Git Directory

This is the remote repository, after pushing the changes then only the modified or updated will be reflected in git directory.

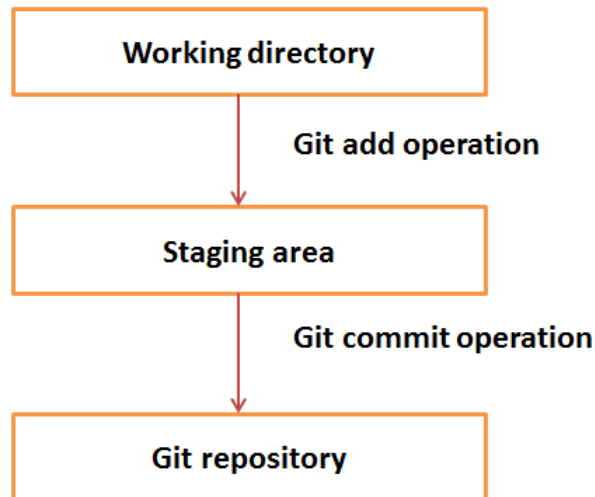


Figure 5.3: Basic Flow of Git

5.2.1 Basic Commands

- git clone: It is used to checkout the project.
- git add: When new files are added in local, then to add in staging area.
- git config: use to configure the username and email.
- git push: It is used to push all committed changes in the remote repository.
- git pull: Pull any changes from origin.
- git commit : It will the files with commit message only in staging area.

The basic commands are all shown in the given figures

```

Home@DESKTOP-UCLHDTQ MINGW64 /d/git practice
$ git config --global user.name "shubhrasinghal"

Home@DESKTOP-UCLHDTQ MINGW64 /d/git practice
$ git config --global user.email "shubhrasinghal2015@gmail.com"

Home@DESKTOP-UCLHDTQ MINGW64 /d/git practice
$ git init
Initialized empty Git repository in D:/git practice/.git/

Home@DESKTOP-UCLHDTQ MINGW64 /d/git practice (master)
$ git clone https://github.com/coderss21/JavaPractice.git
Cloning into 'JavaPractice'...
remote: Enumerating objects: 27, done.
remote: Counting objects: 100% (27/27), done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 27 (delta 4), reused 19 (delta 2), pack-reused 0
Receiving objects: 100% (27/27), 4.31 KiB | 275.00 KiB/s, done.
Resolving deltas: 100% (4/4), done.

```

Figure 5.4: Basic Command Git 1

```

Home@DESKTOP-UCLHDTQ MINGW64 /d/git practice/JavaPractice (master)
$ ll
total 6
-rw-r--r-- 1 Home 197121 438 May  2 15:33 JavaPractice.iml
drwxr-xr-x 1 Home 197121  0 May  2 15:43 LeetCode/
drwxr-xr-x 1 Home 197121  0 May  2 15:43 out/
-rw-r--r-- 1 Home 197121  88 May  2 15:12 README.md

Home@DESKTOP-UCLHDTQ MINGW64 /d/git practice/JavaPractice (master)
$ cd LeetCode/

Home@DESKTOP-UCLHDTQ MINGW64 /d/git practice/JavaPractice/LeetCode (master)
$ ll
total 24
-rw-r--r-- 1 Home 197121 1043 May  2 15:43 AddTwoNumbers.java
drwxr-xr-x 1 Home 197121  0 May  2 15:23 gradle/
-rwxr-xr-x 1 Home 197121 5296 May  2 15:23 gradlew*
-rw-r--r-- 1 Home 197121 2260 May  2 15:23 gradlew.bat
-rw-r--r-- 1 Home 197121 1134 May  2 15:12 PrintInOrder.java
-rw-r--r-- 1 Home 197121 1108 May  2 15:36 TwoSumProblem.java

Home@DESKTOP-UCLHDTQ MINGW64 /d/git practice/JavaPractice/LeetCode (master)
$ git add AddTwoNumbers.java

Home@DESKTOP-UCLHDTQ MINGW64 /d/git practice/JavaPractice/LeetCode (master)
$ git add *
warning: LF will be replaced by CRLF in LeetCode/gradle/wrapper/gradle-wrapper.p
roperties.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in LeetCode/gradlew.
The file will have its original line endings in your working directory

Home@DESKTOP-UCLHDTQ MINGW64 /d/git practice/JavaPractice/LeetCode (master)
$ git commit -m "Add two numbers leet code problem"
[master c14b126] Add two numbers leet code problem
6 files changed, 287 insertions(+)
create mode 100644 LeetCode/AddTwoNumbers.java
create mode 100644 LeetCode/gradle/wrapper/gradle-wrapper.jar
create mode 100644 LeetCode/gradle/wrapper/gradle-wrapper.properties
create mode 100644 LeetCode/gradlew
create mode 100644 LeetCode/gradlew.bat

```

Figure 5.5: Basic Command Git 2

```
Home@DESKTOP-UCLHDTQ MINGW64 /d/git practice/JavaPractice/LeetCode (master)
$ git log --since=2.days
commit c14b126f728d54c32ad8a7738e22954ae6a9f99f (HEAD -> master)
Author: shubhrasinghal <shubhrasinghal2015@gmail.com>
Date: Sat May 2 15:49:29 2020 +0530

    Add two numbers leet code problem

Home@DESKTOP-UCLHDTQ MINGW64 /d/git practice/JavaPractice/LeetCode (master)
$ git log --since=2.days --oneline
c14b126 (HEAD -> master) Add two numbers leet code problem
```

Figure 5.6: Basic Command Git 3

```
Home@DESKTOP-UCLHDTQ MINGW64 /d/git practice/JavaPractice/LeetCode (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  ../.idea/
  ../JavaPractice.iml
  .gradle/
  .idea/
  ../out/

nothing added to commit but untracked files present (use "git add" to track)

Home@DESKTOP-UCLHDTQ MINGW64 /d/git practice/JavaPractice/LeetCode (master)
$ git push origin
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 4 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (11/11), 53.48 KiB | 4.86 MiB/s, done.
Total 11 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/coderss21/JavaPractice.git
   49a781c..c14b126  master -> master

Home@DESKTOP-UCLHDTQ MINGW64 /d/git practice/JavaPractice/LeetCode (master)
$ git pull origin
Already up to date.
```

Figure 5.7: Basic Command Git 4

Chapter 6

Implementation & Phase 2nd of Project Work

6.1 Multi-threading

First my code is working on single thread so to run a big process on single thread is time consuming process. So to improve the efficiency and reduce the storing and parsing time we introduced multi threading environment. This multi-threading has greatly reduced the time by around 60% compare to before when we were using single thread.

The fig shown in 6.1 is describing the process how do I applied multi-threading structure

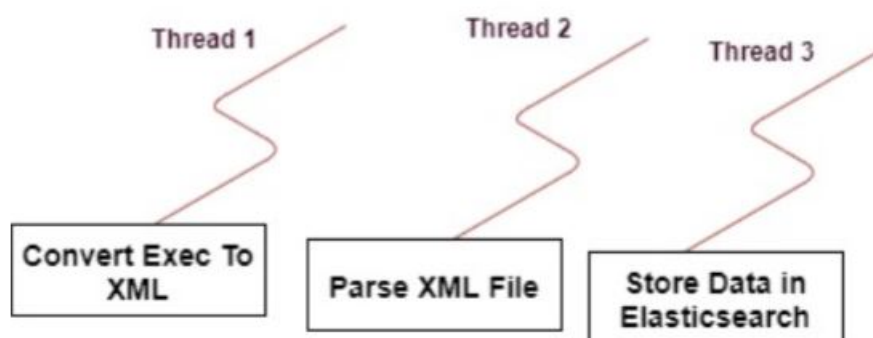


Figure 6.1: Working of Multi-threading

in our code. Multi-threading is a very efficient approach in Java which i applied in this manner within the code.

The first thread will convert the exec file to the XML file and once done then second thread will start parsing that XML file and the first thread will be working on some other

Exec file. When second thread parse the XML then after this third thread will be Storing data in elastic search and once data is stored that XML file will be deleted from machine and second thread will be working in another XML file. This process will continue until the folder with exec file and then with XML file will not be empty. In this way we are saving lot of space as by removing the files after data is stored in Elastic search. **So multi-threading will reduce the time and space.**

As to fetching the Exec file is the incremental process because if there are some new class added in the project which was never there then we need to copy files. On the second if we run this code then it is required to keep track that only new exec data will be pushed in elastic search. So for this process we will maintain another elastic search database which will keep record of only class name, so when we will run code again for some project then that whole process will be repeated only for those changed. Otherwise the Elastic search database will be remain same.

6.2 Perl Script for Getting Exec File From Remote Server and Untar the latest Build

In the first phase I write a Perl script to generating the Exec file. Now for the second phase of project we need to copy the exec file from remote server and also need to deploy a new build before doing any further process in the code. So instead of writing the Linux commands daily, we written a Perl script. In this Perl Script we use the basic and important Linux commands. This Perl Scripts is consisting of these main steps:

Step 1: Copying the Exec files from remote server to local server.

Step 2: Untar the latest build of product for which we copied the Exec files.

Step 3: Taking the jar from the build and copied to specified location.

Step 4: Unjar the jar of product and remove unnecessary files which are not required.

Step 5: After removing unnecessary files the modified jar of product is created.

The untar and creating modified jar of product is accomplished to do the further process.

To generate XML files we need the .class files and the latest build

6.3 Description of second Phase

In this phase we are fetching the projects from git to our local repository and then running the git log to fetch history of latest commit and storing this git history into the text file.

Then parsing the text file to get the class name and finally fetch the test name mapped with class from elastic search that we have stored already. This complete flow is explained in block diagram shown in fig 6.2.

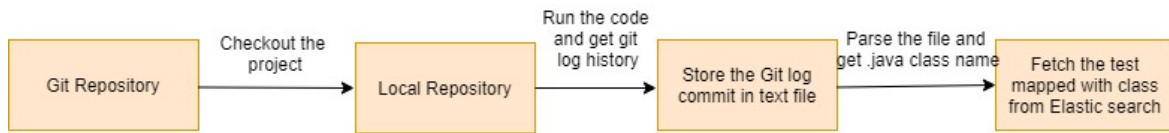


Figure 6.2: Block Diagram

6.3.1 Explanation of Each Block in Flow Diagram

In the 1st phase we took the exec file of each product and after converting and parsing into XML we store the class name mapped with the test into Elastic search engine. In this phase we will first take the latest commits of the code from git and then from elastic search engine we will fetch the test name mapped with the committed classes. The brief description of each part of the phase is explained here:

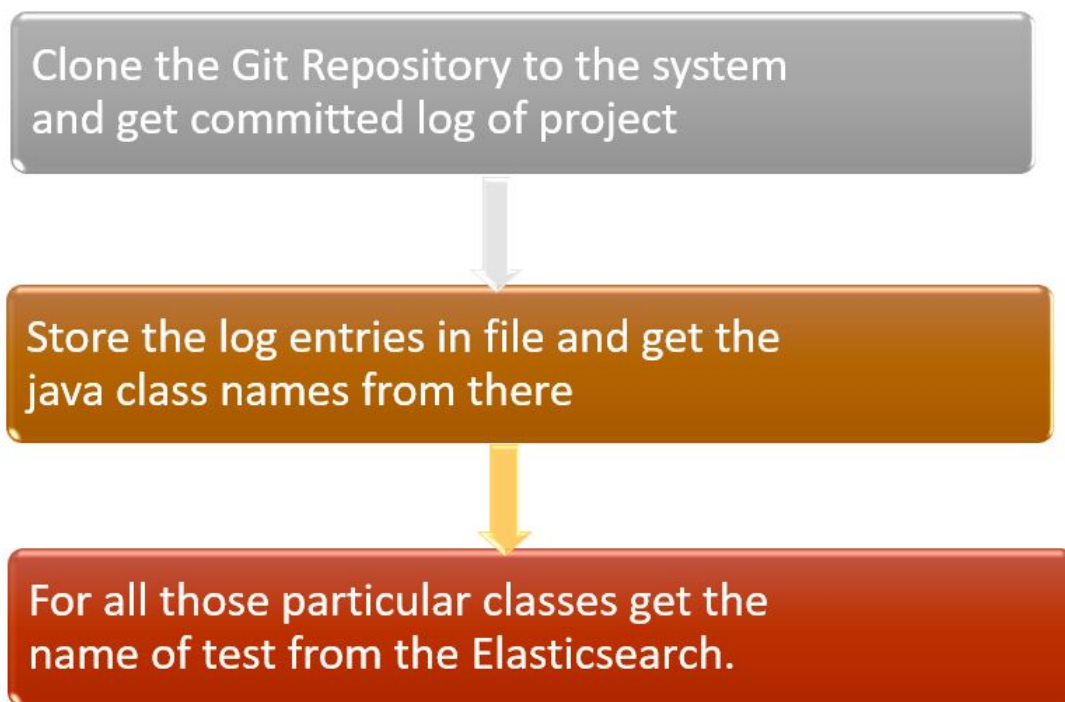


Figure 6.3: Flow Diagram of 2nd Phase

1. Clone the Git Repository to the System and get Committed logs of Project

In this part if the code is not in our local machine then first we will clone the project from the git remote repository. The command use to remote the clone repository.

git clone “HTTP_URL”

If the code is already there, then we will only pull the changes so that if there will be any commit on the code by any developer of that product it can be reflect in our local also. The command to pull the git changes.

git pull “HTTP_URL” So after pulling or cloning the product code we will check the git log for like 2 days. The command to check the git log entries.

git log –since = 2.days

The git log for the particular project will be check , the project whose name will be passed the all actions will be performed on that.

2. Store the log entries in file and get the name of java class from File

The git log which was generated from git need to be store in some text file. After storing the git log we need to parse that file because it contains so many other information which is not required for us, like the name of author, date and other things. So we need to parse that file and get the files which contains **.java** extension.

3. For all those particular classes get the name of test from Elastic search

In this part the output that we generated from the previous one that is the name of java class files which are the new committed class so we will fetch the test name from the elastic search that we have stored in elastic search previously.

Chapter 7

Verification and Validation of Software

7.1 Validation & Verification

Before releasing the product, for quality assurance each product need to go through with verification and validation phase. In **Verification** process we check that the software is upto its specification and building the product correctly. In verification we check that whether we getting the ouput as per input or not. While in **Validation** process we check that the software must do what it is intend for and it means we building the product right. In this we check that software will be accepted by user or not. There is a thin line difference between these two things but this is the most important and crucial part of any good software.

When we are developing a software then it is good practice to apply V & V process on each and every phase of the software process. In doing so we establish a confidence whether software is fit for process further stage or not. It doesn't mean that it is free of defects but it builds a confidence that it is good enough for the intended use.

Methods for Verification:

Review, Walkthrough, Inspection

Methods for Validation:

Testing, End Users

7.2 Methods of Verification

7.2.1 Review

It is an organized way of examining the files consisting of design specs, requirements specifications, code, etc. By one or a couple of individual, to explore defects in the software.

7.2.2 Walkthrough

It is, commonly an informal way of comparing the software product, in which special teams, whether or not related or non-associated with the software program improvement, is going via the software product and feature discussion on feasible mistakes and different defects, presents inside the software program.

7.2.3 Inspection

It is one of the preferred and maximum common strategies of the static trying out. Unlike, walkthroughs, it is a proper way of testing the software program, via exam of files, accomplished by the professional moderator. This technique normally, checklist, guidelines, access and exist criteria at the side of coaching and sharing of reports, as a way to take corrective vital movements.

7.3 Methods of Validation

7.3.1 Testing

By writing the test cases and automate test scenarios we perform the testing, unit testing , functional testing. It helps to validate that all the functions are working what they meant to be.

7.3.2 End Users

Once when the product is ready to deliver than the End users will be one who do the validation by checking the functionality. That the product must be working as same as they mentioned in specification. If the product is successfully work as per the end user requirement. Then we can say we build the product correct.

Chapter 8

Conclusion

The class name mapped with test name is being pushed into elastic search. Working with multi-threads to dump the data into elastic search gave a better performance than using a single thread. The git log is used to get the latest commit history and from that only, the class names of the recent changes are fetched. These class names are then used to fetch test name which was stored in elastic search and then run the test cases. This ensures the priority of recent changes in the codebase to be evaluated.

By applying this approach we are able to achieve a reduction in regression testing time. Now the test case report is generated in real-time during the execution of the nightly regression on a new build. This reduces the validation time of new build which in turn reduces the development time. There is an option available to run all the existing and if any new test case added in real-time to know the working of the product. In this way, we reduced the time as a developer doesn't have to wait to finish the whole regression and can get result effectively with no time.

Bibliography

- [1] Git documentation.
- [2] Jacoco Command Line Interface, <https://www.jacoco.org/jacoco/trunk/doc/cli.html>.
- [3] Optimized Regression Test Using Test Case Prioritization. *Procedia Computer Science*, 79:152–160, 2016.
- [4] Elasticsearch-quick guide, 2019.
- [5] Anonymous. What is regression testing? definition, tools, method, and example, 2019.
- [6] Ahlam S.A. Ansari, Kailas K. Devadkar, and Prachi Gharpure. Optimization of test suite-test case in regression test. *2013 IEEE International Conference on Computational Intelligence and Computing Research, IEEE ICCIC 2013*, pages 1–4, 2013.
- [7] Anshudeep. HashSet in Java With Examples, <https://netjs.blogspot.com/2018/07/hashset-in-java.html>, 2018.
- [8] guru 99. Elk stack tutorial: Learn elasticsearch, logstash, and kibana, 2019.
- [9] C. P. Indumathi and S. Madhumathi. Cost aware test suite reduction algorithm for regression testing. *Proceedings - International Conference on Trends in Electronics and Informatics, ICEI 2017*, 2018.
- [10] Janhavi and Ashima Singh. Efficient regression test selection and recommendation approach for component based software. *Proceedings of the 2014 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2014*, pages 1547–1553, 2014.

- [11] Jackson Joseraj. JVM Architecture Explained,<https://dzone.com/articles/jvm-architecture-explained>, 2016.
- [12] Dusica Marijan and Marius Liaaen. Practical selective regression testing with effective redundancy in interleaved tests. *Proceedings - International Conference on Software Engineering*, pages 153–162, 2018.
- [13] Sudhir Kumar Mohapatra and Manoranjan Pradhan. Finding representative test suit for test case reduction in regression testing. *IEEE International Conference on Computer Communication and Control, IC4 2015*, pages 1–6, 2016.
- [14] Saurabh. What is jenkins? jenkins for continuous integration, 2019.