Study and Analysis of Automation Testing Techniques

Shweta Yadav [#1], Mrs Mamta Yadav [#2]
M.Tech Scholar CSE, Assitant Professor CSE
YCET, Narnaul (India)

Abstract: *Testing is a very important activity in Software Development Process. It is to examine & modify source code. Effective Testing produces high quality software. This Paper deals with a significant and vital issue of Software Testing. Testing can be conducted manually as well as Automated. These Techniques have their own advantages & disadvantages. The Objective of this paper is to perform Automation Testing using Software Testing Tool "Selenium". With this web testing tool, test cases are automatically recorded in background while tester is entering the data in a web application screen.*

Keywords — *Automation Testings, Software Testing, Selenium, Testing principles, Testing Limitations.*

I. INTRODUCTION-

The term software engineering was proposed in 1968 after the discussion of 'software crises for new methods and techniques. From 1968, the development of software engineering has improved our software and effective methods of software specifications, design and implementation. Software engineering includes many meaningful terms:

Software -The set or collections of programs is known as software. Software products may be developed for a particular customer or may be developed for a general market.

Software engineering- Software engineering is an engineering which is concerned with all respects of software production whose aim is the production of quality software that is delivered on time, within budget and that satisfies its requirements. The aim of software engineering is to make the software fault free software and satisfies user's needs.

Software process- A set of activities whose goal is the development of software.

Software process models- A simplified graphical representation of a software process, from the development phase to maintained phase. Generally all process models depend upon the software development life cycle.

Feasibility study

Requirement analysis and specification

- Design
- Coding
- Testing
- Maintenance

Costs of software engineering – roughly 60% of costs are development costs, 40% are testing costs. The overview of software engineering:

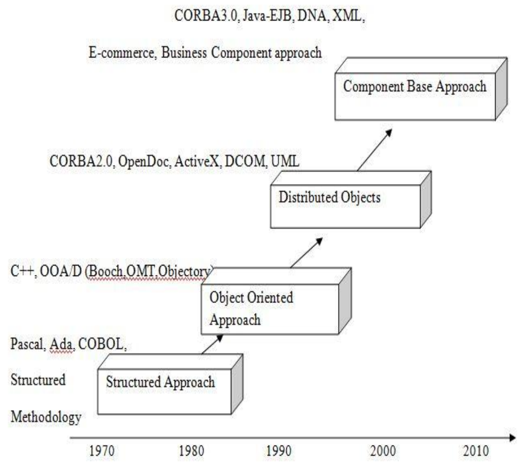


Figure 1.1 the evolution to components in the industry

A. TESTING IN SOFTWARE ENGINEERING

Software testing is any activity that checks or evaluates the software and determining that it meets its required results. Testing is more than just debugging. The purpose of testing can be quality assurance, verification and validation, or reliability estimation. Testing can be used as a generic metric as well. Correctness testing and reliability testing are two major areas of testing. Software testing is a trade-off between budget, time and quality.

The aim of the testing process is to identify all defects existing in a software product. Testing is a program consisting of subjecting the program to a set of test inputs or test cases and observing if the program behaves as expected. If the program fails to behave as expected, then the conditions under which failure occurs are noted for later debugging and correction. The following are the some commonly used terms associated with testing

- Error - An error is a mistake or a bug that refers to the difference between the actual output of software and the correct output. Error is a major of the difference between the actual and the ideal. Error is also used to refer human action that results in software containing a defect or fault.

- Fault - Fault is defect or a condition that causes a system to fail in performing its required function. Fault is taking a variation of error.
- Failure - Failure is the inability of a system to perform a required function according to its specification. A software failure occurs if the behavior of the software is different from the specified behavior. Failure may be caused due to functional or performance reason. A failure is produced only when there is a fault in the system.

B. TYPES OF TESTING

- Unit Testing: Software verification and validation method in which a programmer tests if individual units of source code are fit for use. It is usually conducted by the development team
- Integration Testing: The phase in software testing in which individual software modules are combined and tested as a group. It is usually conducted by testing teams.

Component Testing: Testing technique similar to unit testing but with a higher level of integration - testing is done in the context of the application instead of just directly testing a specific method. Can be performed by testing or development teams.

System Testing: The process of testing an integrated hardware and software system to verify that the system meets its specified requirements. It is conducted by the testing teams in both development and target environment.

Static Testing: A form of software testing where the software isn't actually used it checks mainly for the sanity of the code, algorithm, or document. It is used by the developer who wrote the code.

Dynamic Testing: Term used in software engineering to describe the testing of the dynamic behavior of code. It is typically performed by testing teams.

Black box Testing: A method of software testing that verifies the functionality of an application without having specific knowledge of the application's code/internal structure. Tests are based on requirements and functionality. It is performed by QA teams.

Functional Testing: Type of black box testing that bases its test cases on the specifications of the software component under test. It is performed by testing teams.

White box Testing: Testing technique based on knowledge of the internal logic of an application’s code and includes tests like coverage of code statements, branches, paths, conditions. It is performed by software developers.

Requirements Testing: Testing technique which validates that the requirements are correct, complete, unambiguous, and logically consistent and allows designing a necessary and sufficient set of test cases from those requirements. It is performed by QA teams.

- Security Testing: A process to determine that an information system protects data and maintains functionality as intended. It can be performed by testing teams or by specialized security-testing companies.
- Recovery Testing: Testing technique which evaluates how well a system recovers from crashes, hardware failures, or other catastrophic problems. It is performed by the testing teams.
- Performance Testing: Functional testing conducted to evaluate the compliance of a system or component with specified performance requirements. It is usually conducted by the performance engineer.
- Alpha Testing: Type of testing a software product or system conducted at the developer's site. Usually it is performed by the end user.
- Beta Testing: Final testing before releasing application for commercial purpose. It is typically done by end-users or others.
- Acceptance Testing: Formal testing conducted to determine whether or not a system satisfies its acceptance criteria and to enable the customer to determine whether or not to accept the system. It is usually performed by the customer

II. COMPONENT BASED SOFTWARE ENGINEERING

Component based Software Engineering is the most common term nowadays in the field of software development. Component-Based Software Engineering (CBSE) is concerned with composing, selecting and designing components. As the popularity of this term and number of commercially available software components grows, selecting a set of components to satisfy a set of requirements while lowest cost is becoming more difficult. The basic idea is to develop software systems by selecting appropriate off-the-shelf components and then to assemble them with a well-defined software architecture.

Component-based development (CBD) is a branch of software engineering that indicate the separation of the wide-ranging functionality available throughout a given software system. It is a reuse-based approach to defining, implementing and composing loosely coupled independent components into systems. This aims to bring about an equally wide-ranging degree of benefits in both the short-term and the long-term for the software itself and for organizations that sponsor such software.

This new software development approach is very different from the traditional approach. These commercial off-the shelf (COTS) components can be developed by different developers using different languages and different platforms.

where COTS components can be checked out from a component repository, and assembled into a target software system. Component-based software development (CBSD) can effectively reduce development cost and time-to-market, and improve maintainability, reliability and overall quality of software systems.

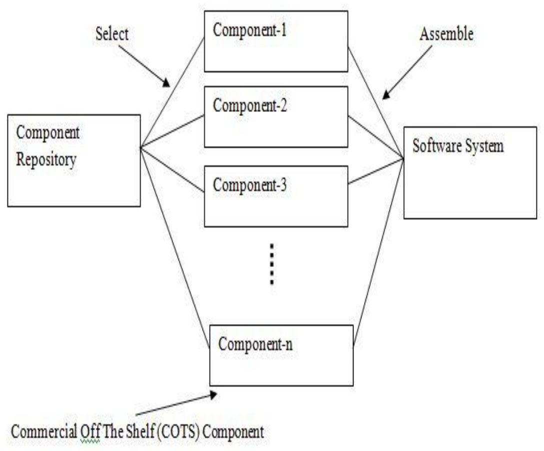


Figure 1.2: Component-Based Software Development

III. USE OF JUNIT &ECLIPSE

JUnit is a framework for implementing testing in Java. It provides a simple way to explicitly test specific areas of a Java program, it is extensible and can be employed to test a hierarchy of program code either singularly or as multiple units

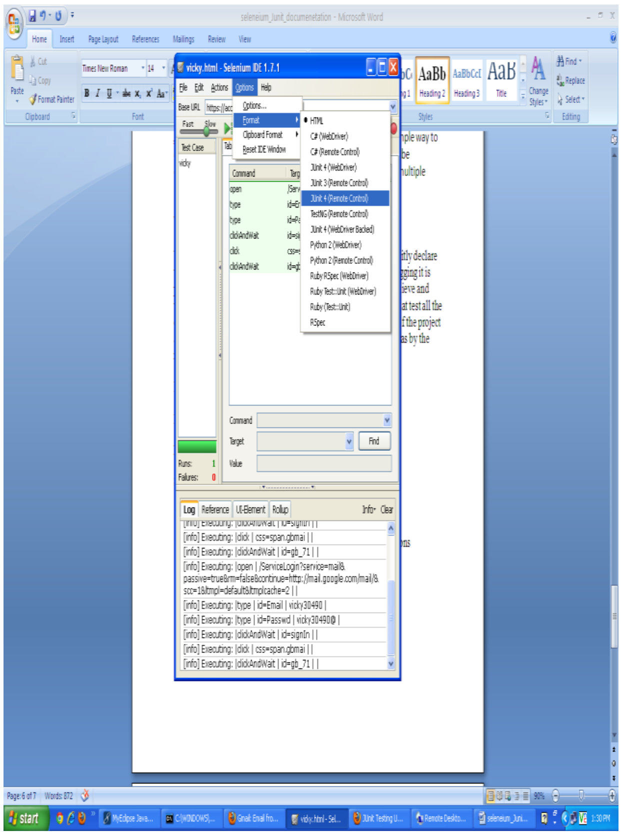
Making a Test Case in Junit:

Step 1:open myeclipse & create a web Project.

Step2: Create a class file in that project.

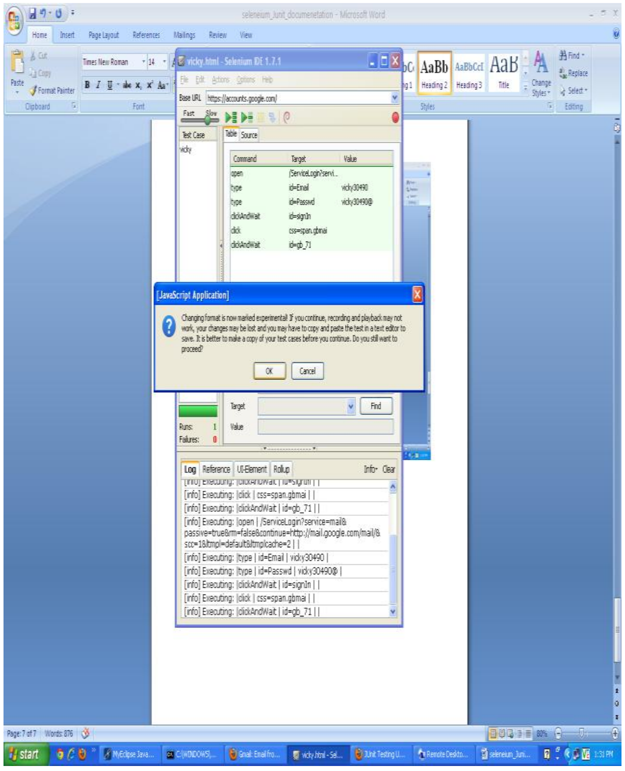
Step3: Now go to selenium IDE And Open that Test Case .Go to Options Select

Formats->Junit4(Remote Control)



```
[info] Executing: |click | css=span.gbmai | |
[info] Executing: |clickAndWait | id=gb_71 | |
[info] Executing: |open | /ServiceLogin?service=mail&
passive=true&rm=false&continue=http://mail.google.com/mail/&
scc=1&ltmpl=default&ltmplcache=2 | |
[info] Executing: |type | id=Email | vicky30490 |
[info] Executing: |type | id=Passwd | vicky30490@ |
[info] Executing: |clickAndWait | id=signIn | |
[info] Executing: |click | css=span.gbmai | |
[info] Executing: |clickAndWait | id=gb_71 | |
```

Step4: Select ok


vicky.html - Selenium IDE 1.7.1
[JavaScript Application]
OK
Cancel
Log Reference UI-Element Rollup Info Clear

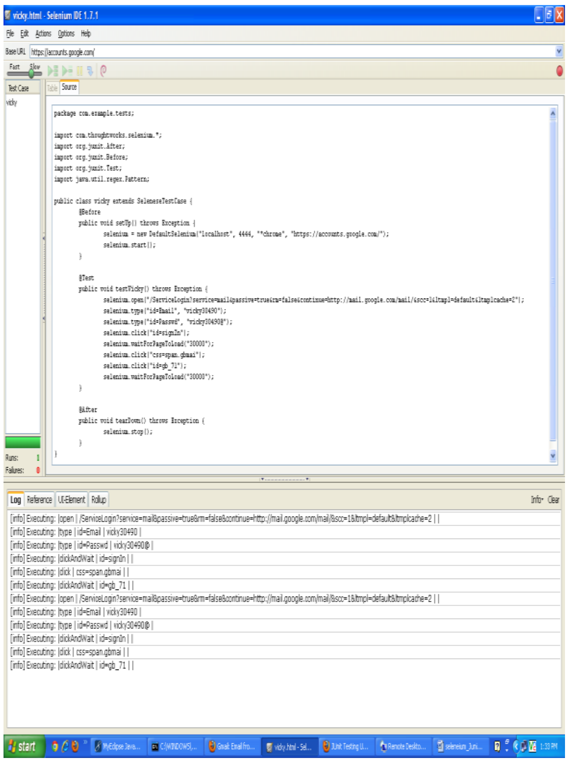
Base URL: https://accounts.google.com/

Command	Target	Value
open	/ServiceLogin?servi...	
type	id=Email	vicky30490
type	id=Passwd	vicky30490@
clickAndWait	id=signIn	
click	css=span.gbmai	
clickAndWait	id=gb_71	

Changing format is now marked experimental! If you continue, recording and playback may not work, your changes may be lost and you may have to copy and paste the test in a text editor to save. It is better to make a copy of your test cases before you continue. Do you still want to proceed?

```
[info] Executing: |click | css=span.gbmai | |
[info] Executing: |clickAndWait | id=gb_71 | |
[info] Executing: |open | /ServiceLogin?service=mail&
passive=true&rm=false&continue=http://mail.google.com/mail/&
scc=1&ltmpl=default&ltmplcache=2 | |
[info] Executing: |type | id=Email | vicky30490 |
[info] Executing: |type | id=Passwd | vicky30490@ |
[info] Executing: |clickAndWait | id=signIn | |
[info] Executing: |click | css=span.gbmai | |
[info] Executing: |clickAndWait | id=gb_71 | |
```

Step6: It will convert the whole test case in a Junit java code .



```
package com.example.tests;

import com.thoughtworks.selenium.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import java.util.regex.Pattern;

public class vicky extends SeleneseTestCase {
        @Before
        public void setUp() throws Exception {
                selenium = new DefaultSelenium("localhost", 4444, "*chrome", "https://accounts.google.com/");
                selenium.start();
        }

        @Test
        public void testVicky() throws Exception {
                selenium.open("/ServiceLogin?service=mail&passive=true&rm=false&continue=http://mail.google.com/mail/&scc=1&ltmpl=default&ltmplcache=2");
                selenium.type("id=Email", "vicky30490");
                selenium.type("id=Passwd", "vicky30490@");
                selenium.click("id=signIn");
                selenium.waitForPageToLoad("30000");
                selenium.click("css=span.gbmai");
                selenium.click("id=gb_71");
                selenium.waitForPageToLoad("30000");
        }

        @After
        public void tearDown() throws Exception {
                selenium.stop();
        }
}
```

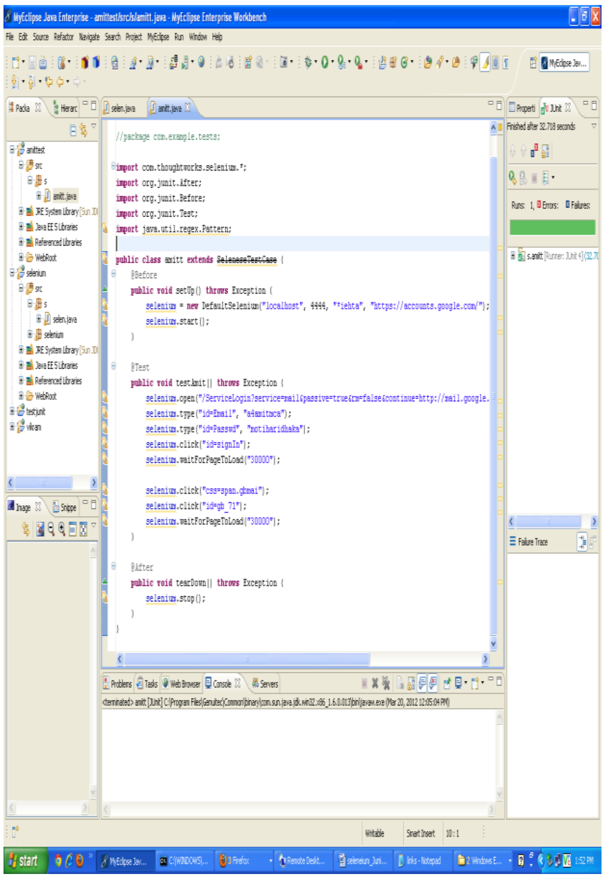
```
[info] Executing: |open | /ServiceLogin?service=mail&passive=true&rm=false&continue=http://mail.google.com/mail/&scc=1&ltmpl=default&ltmplcache=2 | |
[info] Executing: |type | id=Email | vicky30490 |
[info] Executing: |type | id=Passwd | vicky30490@ |
[info] Executing: |clickAndWait | id=signIn | |
[info] Executing: |click | css=span.gbmai | |
[info] Executing: |clickAndWait | id=gb_71 | |
[info] Executing: |open | /ServiceLogin?service=mail&passive=true&rm=false&continue=http://mail.google.com/mail/&scc=1&ltmpl=default&ltmplcache=2 | |
[info] Executing: |type | id=Email | vicky30490 |
[info] Executing: |type | id=Passwd | vicky30490@ |
[info] Executing: |clickAndWait | id=signIn | |
[info] Executing: |click | css=span.gbmai | |
[info] Executing: |clickAndWait | id=gb_71 | |
```

vicky.html - Selenium IDE 1.7.1
File Edit Actions Options Help
Base URL https://accounts.google.com/
Test Case
Source
Runs: 1
Failures: 0
Log Reference UI-Element Rollup

Step7:Copy the code and Paste into the MyEclipse class & save it .
Step8 :It will show a lot of errors so we need to remove them .
Step9: First of all download jar files for this junit
We need 2 jar files

1. Junit-4.8.1.jar(Junit Annotations)
2. Selenium-server-standalone-2.5.0.jar(ALL Commands)

Link:
http://code.google.com/p/selenium/downloads/detail?name=selenium-server-standalone-2.20.0.jar&can=2&q=
link:
http://grepcode.com/snapshot/repo1.maven.org/maven2/junit/junit/4.8.1
Step10: Go to project ->properties->java Build Path->Libraries->Add external Jar Files.
Step11: Almost All errors Get Corrected.
Step12: Now Run As Junit Test.
Step13: It will an error server now connected .so open cmd And type command Java -jar selenium-server-standalone-2.5.0.jar –port 4444
Step14: Now server Will run in background & when Run wait for 2 windows of IE to be Opened .
Step15: If run Successful then green line is Shown like-



```
//package com.example.tests;

import com.thoughtworks.selenium.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import java.util.regex.Pattern;

public class amitt extends SeleneseTestCase {
    @Before
    public void setUp() throws Exception {
        selenium = new DefaultSelenium("localhost", 4444, "*iehta", "https://accounts.google.com/");
        selenium.start();
    }

    @Test
    public void testAmit() throws Exception {
        selenium.open("/ServiceLogin?service=mail&passive=true&rm=false&continue=http://mail.google.
        selenium.type("id=Email", "a4amitmca");
        selenium.type("id=Passwd", "motiharidhaka");
        selenium.click("id=signIn");
        selenium.waitForPageToLoad("30000");

        selenium.click("css=span.gbmai");
        selenium.click("id=gb_71");
        selenium.waitForPageToLoad("30000");
    }

    @After
    public void tearDown() throws Exception {
        selenium.stop();
    }
}
```

```
<terminated> amitt [JUnit] C:\Program Files\Genuitec\Common\binary\com.sun.java.jdk.win32.x86_1.6.0.013\bin\javaw.exe (Mar 20, 2012 12:05:04 PM)
```

MyEclipse Java Enterprise - amittest/src/s/amitt.java - MyEclipse Enterprise Workbench
File Edit Source Refactor Navigate Search Project MyEclipse Run Window Help
selen.java
amitt.java
Finished after 32.718 seconds
Runs: 1
Errors:
Failures:
Failure Trace
Problems Tasks Web Browser Console Servers
Writable Smart Insert

NOTE:- Sometimes You get error so in coding change browser
Select IE browser by Replacing *chrome by *iehta .
i.e –Selenium = new DefaultSelenium(“localhost”,4444,”*iehta”,” https://accounts.google.com)

IV. CONCLUSION

Our target is to create a feasible Automatic testing tool with minimal human involvement and significant performance improvement. Our complete system would provide almost complete automation to the tester. Good coverage is an important criterion and efficient Conformance testing provides significantly better coverage than other techniques.

REFERENCES

[1] PRESSMAN Roger, Software Engineering, McGraw Hill, 1997
[2] HERZUM Peter, SIMS Oliver, Business Component Factory, Wiley, 1999
[3] HUIZING M., Component Based Development, http://www.win.tue.nl/xootic/magazine/jan-1999/huizing.pdf,last access:01.12.2005
[4] WANG Ju An, Towards Component-Based Software Engineering,wang.pdf?key1=357729&key2=7337455011&coll=GUIDE&dl=GUIDE&CFID=36365381&CFTOKEN=91792709, last access:01.12.2005
[5] HILL, Bennett, McROBB, Farmer, Object Oriented System Analysis and Design (using UML), 2nd Edition, McGraw Hill, 2002