**Topics in Software Engineering**

**Mid Term Assignment**

****

**Submitted by**

**Rizwan Mukhtiar**

MS (Computer Science)

Registration ID: 5952

**Assigned by**

**Dr. Fazal-E-Malik**

**Department of Computer Science**

**Faculty of Engineering and Applied Sciences**

**Iqra National University, Peshawar**

**Q.1 a) What is GUI Testing?**

**b) Study open source tools available to conduct GUI Testing and write notes on**

**only two tools**

1. **What is GUI?**

The GUI stands for **Graphical User Interface**.it is a form of user interface that allows users to interact with electronic devices or software system through graphical icons or elements and type text base to interact with the system.

**GUI Testing:**

It is the process of testing the graphical interface of the system having application is under test. Simple GUI Testing is a process of testing the application's graphical user interface to ensure proper functionality as per the specifications. It involves checking the application components like buttons, icons, checkboxes, color, menu, toolbar, dialog boxes, windows etc.

For example if you visit any website or a software application then first look will be the page of graphical user interface of the site. User simple does not have any knowledge on the source code. Testing just involves the design structure, images that are working properly or not. And these images should be completely visible in different browser or application. Links are properly working or not, available buttons in the application or website are working or not when clicking on these buttons. And if the user wants to minimize or resize the screen then neither images nor contents should shrink or copy.

1. **Open Source Tools for GUI Testing:**

Any application or website can be considered good if it is user-friendly and easy to manage. But the very first thing that catches user’s attention is the look and feel of the application i.e. (Graphical User Interface). Hence GUI Testing becomes important to remove gaps in the design and gain users attraction towards the system. In today’s era of digitization, GUT Testing is not limited up to desktop computers, it is expanding its edges on smartphones and tablets like electronic devices.

We can automate GUI testing by using different GUI testing tools. A list of open source GUI testing tools are as follows:

# Ranorex Studio, CrossBrowserTesting, CubicTest, eggplant, Ascentialtest, iMacros, RIATest, SilkTest, Squish, Selenium, TestComplete, TestPartner, Jubula, IcuTest, Sahi, Telerik Test, Studio, Cucumber, LoadUI, Watir, Espresso, Katalon Studio etc.

**Notes on Two Tools:**

As discussed above we have multiple tools available but here we are discussing note on the following two of them.

1. **Selenium:**

SELENIUM is a free (open-source) automated testing framework used to validate web applications across different browsers and platforms. You can use multiple programming languages like Java, C#, Python etc. to create Selenium Test Scripts. Testing done using the Selenium tool is usually referred to as Selenium Testing. It is a portable [framework](https://en.wikipedia.org/wiki/Software_framework) for [testing](https://en.wikipedia.org/wiki/Software_testing) [web applications](https://en.wikipedia.org/wiki/Web_application). Selenium provides a playback tool for authoring [functional tests](https://en.wikipedia.org/wiki/Functional_testing) without the need to learn a test [scripting language](https://en.wikipedia.org/wiki/Scripting_language). Selenium is the household name when it comes to test automation. It is considered the industry standard for user interface automation testing of Web applications. Almost nine out of ten testers are using or have ever used Selenium in their projects, according to [Survey on Test Automation Challenges](https://www.katalon.com/resources-center/blog/infographic-challenges-test-automation/).

For developers and testers who have experience and skills in programming and scripting, Selenium offers flexibility that is unseen in many other test automation tools and frameworks. Users can write test scripts in many different languages (such as Java, Groovy, Python, C#, PHP, Ruby, and Perl) that run on multiple system environments (Windows, Mac, Linux) and browsers (Chrome, Firefox, IE, and Headless browsers).

Selenium has recently released its first alpha version of Selenium in April 2019. The announcement of Selenium 4 official version is yet to be determined; but you can expect the release would come with many improved and enriched features. Selenium Software is not just a single tool but a suite of software, each piece catering to different testing needs of an organization. Here is the list of tools.

Figure 1: Selenium Suite



To use Selenium effectively, users have to possess advanced programming skills and need to spend considerable time to build automation frameworks and libraries necessary for automation. This is a main disadvantage of Selenium, which is addressed in other tools built for codeless test automation like Katalon Studio.

**Website:**<https://www.selenium.dev/>

1. **Katalon Studio:**

Katalon Studio is an automation testing solution developed by Katalon LLC. The software is built on top of the [open-source](https://en.wikipedia.org/wiki/Open-source_software) automation frameworks. It is a powerful and comprehensive automation solution for testing API, Web, mobile, and desktop application testing. It also has a rich feature set for these types of testing and supports multiple platforms including Windows, macOS, and Linux.

Leveraging Selenium and Appium engines, Katalon Studio provides a uniquely integrated environment for testers who find difficulties in integrating and deploying different frameworks and libraries to use Selenium and Appium, as well as those who are already familiar with these engines. Katalon helps you quickly generate automated tests cross-platform. Seamlessly integrate automated tests into your CI/CD pipeline. It generates Centralized reports and quality insights with Katalon TestOps

Katalon Studio was named a March 2019 Gartner Peer Insights Customers’ Choice for Software Test Automation with more than 530 positive reviews, which once again, proves the tool is now one of the biggest players in the market.

**Highlights of the tool includes:**

* A complete feature set for test automation of API/Web services, Web, and mobile applications
* Supports both SOAP and RESTful for API and services testing
* Hundreds of built-in keywords for creating test cases
* Support BDD Cucumber to express test scenario in natural languages
* Can be used for both automated and exploratory testing
* The testing capability can be extended via plugins on [Katalon Store](https://store.katalon.com/) In-depth views of reports on [Katalon TestOps](http://analytics.katalon.com/)

In April 2020, Katalon has released its newest update — Katalon Studio 7.3 — with major enhancements that support scalable projects, solving the Wait issue in Selenium, sharing test artifacts across multiple projects, and more

**Website:** <https://www.katalon.com/>

**Q.2 a) What is Usability Testing?**

**b) Study Usability Test Tools available to conduct Usability Test and write**

**notes on only two tools**

1. **What is usability?**

Usability is part of the broader term “user experience” and refers to the ease of access and/or use of a product or website. Usability describes the level of easy use of a system with which a user gets the desired goal. In information technology, the term is often used in relation to software applications and Web sites, but it can be used in relation to any product that is employed to accomplish a task (for example, a toaster, a car dashboard, or an alarm clock).

**Usability Testing:**

Usability testing refers to evaluating a product or service by testing it with representative users. Typically, during a test, participants will try to complete typical tasks while observers watch, listen and takes notes.  The goal is to identify any usability problems, collect qualitative and quantitative data and determine the participant's satisfaction with the product. Simply Usability testing is a method by which users of a product are asked to perform certain tasks in an effort to measure the product's ease-of-use, task time, and the user's perception of the experience. Usability testing can be done formally, in a usability lab with video cameras, or informally, with paper mock-ups of an application or Web site. Changes are made to the application or site based on the findings of the usability tests. Whether the test is formal or informal, usability test participants are encouraged to think aloud and voice their every opinion. Usability testing is best used in conjunction with user-centered design, a method by which a product is designed according to the needs and specifications of users. This testing is recommended during the initial design phase of SDLC, which gives more visibility on the expectations of the users.

1. **Usability Testing Tools:**

UX design is incomplete without user testing, which is an integral part of the process. It’s very important for you to know how your user interacts with your design so you can create the best user experience that allows visitors to fulfill their tasks and increase conversions. This is where usability testing tools comes into play for the testing of your UX design at an enterprise and programming level. [Usability testing tools](https://dynomapper.com/blog/19-ux/271-usability-testing-tools/#usability-testing-tools) are vital to a website's success. The user experience relies heavily on user testing, as it helps developers to understand how users interact with the design. It will consume too much of your time and money to have and run your own user testing lab, which includes searching for and recruiting your own target consumers and setting up the necessary hardware. Usability testing tools are growing in numbers and features. A number of usability testing tools are available, some of the best testing tools are as follows:

[Userlytics](https://www.userlytics.com/), [UsabilityHub](https://usabilityhub.com/), [Lookback](https://lookback.io/), [Hotjar](https://www.hotjar.com/), [TryMyUI,](https://www.trymyui.com/) [UserTesting,](https://www.usertesting.com/)[Crazy Egg](https://www.crazyegg.com/), [Reflector 3](https://www.airsquirrels.com/reflector), [Optimizely](https://www.optimizely.com/), [Inspectlet,](https://www.inspectlet.com/)[QUALAROO](https://qualaroo.com/), [USABILLA](https://usabilla.com/), [FEEDBACK ARMY](http://www.feedbackarmy.com/), [USERFEEL](https://www.userfeel.com/), [ETHNIO](https://ethn.io/), Loop11, UsabilityHub, TestRail, ClickTale, Optimal Workshop and much more.

**Notes on Two Tools:**

As discussed above that we have a lot of usability testing tools are available but here we discussing only two of them.

1. **Optimizely:**

[Optimizely](https://www.optimizely.com/) One of the Best Usability Testing Tools for Your Web App. This testing is one of the most powerful ways to test planned changes on a web page in comparison to the existing design (or an alternative proposed update). It helps determine which version gives the most positive results. Google provides its own A/B testing software known as Google Optimize so you can conduct experiments on your website.

However, a more powerful software tool for A/B testing is Optimizely. The tool provides many easily usable features to assist you in building your tests. On such tool allows you to perform multivariate testing, which is a technique to test many variables on a given web page. You can also create personalized experiences and target particular audiences when you unveil new designs to the visitor (including targeting different URLs, browsers, and geographic regions).

Optimizely turns out to be a comprehensive suite of usability testing tools. But, it is meant for serious businesses, which is shown in the price.

Among the usability testing tools in the market, [Optimizely](https://www.optimizely.com/) is one of the well-known platforms many will recognize. It’s best known feature is its A/B testing, which earned it a spot on our list of top A/B testing tools – it’s other features, however, aren’t bad either!

One of the most attractive traits in this user testing tool is how easy it is to create new studies. There is no coding needed, as users can set everything up right on their dashboard. It is equipped to test modern websites, like pages that have dynamic content.

Figure 2: Optimizely Usability Testing Tool Screenshot.



**Website:** <https://www.optimizely.com/>

1. [**Crazy Egg**](https://www.crazyegg.com/)**:**

[Crazy Egg](https://www.crazyegg.com/) Usability Testing Tool Offer Complete Solution for App Testing. With Crazy Egg usability testing tool, you can know how visitors interact with your website as if with X-ray glasses. This usability testing tool allows you to understand the activity of people on your website, and why visitors are not being converted into regular customers and users.

The objects clicked are identified by heat maps. And, they also identify the sources they are being referred from. One of the most helpful things in Crazy Egg is the scroll map capable of identifying how far visitors scroll down a page before abandoning it completely.

Other helpful things are the ability to monitor the click numbers on each and every element of a page, and to conduct A/B testing. All these things put Crazy Egg on this list.

Crazy Egg is a click-based user experience tool with four main features:

* ‘Heatmap’ logs where each visitor clicks on your webpage.
* ‘Scrollmap’ shows how far down the page each visitor typically scrolls.
* ‘Overlay’ breaks down the number of clicks on each page element.
* ‘Confetti’ provides detailed insights about visitor sources, search terms, and other components

Crazy Egg is, perhaps, most known for their heat maps. And while that is a good feature, we love other aspects of this user testing tool as well. Users praise their Scroll Map feature, which is perfect for knowing at which point in the page you lose participants’ interest. This is quite handy when deciding where the call-to-action should be. Crazy Egg is integrated with Justinmind

Figure 3: Crazy Egg Usability Testing Tool Screenshot.



**Website:** <https://www.crazyegg.com/>

**Q.3 a) What Is Clean room software engineering?**

**b) Study some Case Studies in cleanroom software engineering and write summaries of some case studies**

**Clean Room Software Engineering:**

It is an engineering approach which is used to build correctness in developed software. The main concept behind the cleanroom software engineering is to remove the dependency on the costly processes. The cleanroom software engineering includes the quality approach of writing the code from the beginning of the system and finally gathers into a complete a system.

CSE is a [software development process](https://en.wikipedia.org/wiki/Software_development_process) intended to produce software with a certifiable level of [reliability](https://en.wikipedia.org/wiki/Reliability_engineering). CSE is an approach that highlights the needs of to deploy a software with correctness and to avoid dependencies on cost defect removal processes and verifying correctness before testing. Also on accuracy of software in specification and design, and formal verification of each element of the resultant design is needed.

Cleanroom differs from other formal methods in that it doesn't require mathematically defined requirements—those stated in plain English are adequate. These requirements are divided into tagged statements for traceability. The process of tagging requirements in small verifiable statements allows for tracing and verification of each requirement throughout the process. Moreover, since attempts to document requirements are likely to have errors, inconsistencies, and omissions, Cleanroom refines many of these through the "Box Structure Development Method," a process that treats software as a set of communicating state machines that separate behavioral and implementation concerns

**b) Case Studies in Cleanroom Software Engineering:**

A case study may be understood best as simply a tool for creating an opportunity for conversation. It is a story, told in narrative form and based on actual events. An effective case study transfers specific knowledge and learning by placing participants in a position to think through real-life choices that confronted decision-makers. The element of realism helps participants in the learning process develop analytical techniques they can draw upon when faced with similar choices in their own projects; the narrative context provides a concrete base for practical lessons from the story. Different projects have been performed in cleanroom software engineering. Simply multiple projects are using the cleanroom techniques and processes. Some studies are NASA Satellite Control Project, and the IBM COBOL Structuring Facility.

1) This case study analyzes the application of the [Cleanroom methodology](http://infohost.nmt.edu/~al/cseet-paper.html) to the development of production software at the NASA Goddard Space Flight Center. The Cleanroom methodology emphasizes human discipline in program verification to produce reliable software products that are ‘right the first time.’

Preliminary analysis of the Cleanroom case study shows that the method can be applied successfully in the Flight Dynamics Division environment and may increase staff productivity and product quality. Compared to typical Software Engineering Laboratory activities, there is evidence of lower failure rates, a more complete and consistent set of inline code documentation, a different distribution of phase effort activity, and a different growth profile in terms of lines of code developed.

2) The IBM COBOL Structuring Facility (COBOL/SF) Version 2 Program Product automatically transforms unstructured COBOL programs into structured form. It was developed by a small programming team using Cleanroom Software Engineering technology [Mills 1987) .COBOL/SF Version 2 consists of 80,000 lines (52,000 new and changed over Version 1) of high function source code that was developed under statistical quality control, being specified, then designed, mathematically verified, and coded with no unit debugging in a pipeline of increments at very high productivity. Each increment was placed under engineering change control before any execution and subjected to system test under a sound statistical design. As a result, COBOL/SF passed its field test of structuring a half-million lines of COBOL code in over 300 application programs with only 10 errors detected. All errors were trivial, none requiring more than a few hours to find and fix, and most just a few minutes. In all testing, only one error resulted in a COBOL program failing to execute functionally equivalent before and after structuring. It is understandable that the common wisdom of such a new subject as software engineering can underestimate the potentials of human achievement in various ways. Centuries ago, the common wisdom in arithmetic with Roman numerals was that large scale arithmetic was impractical, so that the great inventory with structural problems caused by complex performed procedure logic, ALTER statements. Etc.

Published reports on software productivity and quality are highly variable, however, averages of 150 WC I person-month and 70 errors I KWC (including unit debugging) are representative of industrial experience for complex products.

**Summary**

In the NASA Satellite Project the total cost of the training of the team was calculated at 4% of the project hours. And the time allocated in the project was 33% of design, 18% of coding, 27% testing and 22% of meetings and other overhead. Some other resources were also utilized, that is 69% of higher productivity, 45% error reduction rate, and 60 to 80 % decreases in resources used.

The IBM COBOL Structuring Facility Program Product was developed by a small programming team using Cleanroom Software Engineering technology in a pipeline of increments with very high quality and productivity. In the Cleanroom approach, programs are developed under statistical quality control. And mathematical verification is used in place of unit debugging. The formal methods of specification, design, functional verification, and testing are described together with development and management practices required for maintaining intellectual control over the process. This took place in 1988. Total ten-fold reduction was carried out in total defects per thousand line of code. And five-fold improvement was performed in developer productivity measured in lines of codes per month. Only seven errors found in the first three years, all of which were simple fixes