GUI Testing Using AI Automation Framework

Submitted By

Dipendra K.Dabhi

20MCEC03

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

MAY 2022

GUI Testing Using AI Automation Framework

Major Project

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology in Computer Science and Engineering

Submitted By

Dipendra K.Dabhi

(20MCEC03)

Guided By

Dr VIJAY UKANI

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

MAY 2022

Certificate

This is to certify that the major project entitled **"GUI Testing Using AI Automation Framework"** submitted by **Dipendra K.Dabhi (20MCEC03)**, towards the partial fulfillment of the requirements for the award of degree of Master of Technology in Computer Science and Engineering of Nirma University, Ahmedabad, is the record of work carried out by him 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 Vijay Ukani
Internal Guide & Associate Professor
CSE Department
Institute of Technology
Nirma University, Ahmedabad

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

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

Dr Rajesh N Patel
Director
Institute of Technology
Nirma University, Ahmedabad

Statement of Originality

I, **Dipendra K.Dabhi**, **20MCEC03**, give undertaking that the Major Project entitled **"GUI Testing Using AI Automation Framework"** 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.

D. K Dabhi

Signature of Student

Date: 17/05/2022

Place:Ahmedabad

Endorsed by

Dr Vijay Ukani

(Signature of Guide)

Acknowledgements

It gives me immense pleasure in expressing thanks and profound gratitude to **Dr Vijay Ukani**, Associate Professor, Computer Science and Engineering Department, Institute of Technology, Nirma University, Ahmedabad for his valuable guidance and continual encouragement throughout this work. The appreciation and continual support he has imparted has been a great motivation to me in reaching a higher goal. His guidance has triggered and nourished my intellectual maturity that I will benefit from, for a long time to come.

It gives me an immense pleasure to thank **Dr Madhuri Bhavsar**, Hon'ble Head of Computer Science And Engineering Department, Institute of Technology, Nirma University, Ahmedabad for her kind support and providing basic infrastructure and healthy research environment.

A special thank you is expressed wholeheartedly to **Dr Rajesh N Patel**, Hon'ble Director, Institute of Technology, Nirma University, Ahmedabad for the unmentionable motivation she has extended throughout course of this work.

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

- Dipendra K.Dabhi

20MCEC03

Abstract

In this research project, I have focused on automation testing, which is done on the web or mobile applications that are based on the graphical user interface (GUI). In the current situation, all the testing that is performed on the web application or the mobile application is done manually. It is very time-consuming and not that accurate to decide whether all the functionality of the web application or mobile application is working properly or not. Buttons, drop-down menus, click, and entered text fields that are necessary are filled properly or not. If the other exceptions that may occur during the interaction with the application are working properly or not. All possible tests are needed to check the quality of the web application or mobile application. Automation testing is a new technique that solves this issue of testing all the functionality of the application by automating all the tests that check the working of the particular module and giving results according to the input given, checking with the already available results, and passing the test case according to this. This will help the organizations to develop the application faster and test that application automatically so the manual efforts can be reduced and get better results. It also gives better results for continuous integration, continuous delivery, and continuous deployment (CICD) approaches.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

In this chapter, we discuss GUI automation testing's importance in software development and quality assurance. The chapter discusses the reasons why automation testing plays an important role in the continuous integration and continuous delivery/continuous deployment (CICD) approach.

1.1 Problem Summary and Introduction

In the current software development life cycle, testing is done at the end of the whole software development life cycle. If we find any bug in the software, then it needs to pass through all the previous steps, which is very time-consuming. And in this faster development, a single bug can also change the major consensus of the application. If the single functionality of an application is not working, then the user can try the other application. To engage the user with the application, the developer must fix bugs and features quickly and provide better quality assurance to the user. To solve this problem, the continuous integration and continuous delivery/continuous deployment (CICD) approaches are used to provide faster deployment of the new features of the application, and continuous testing of the application is done to resolve any bugs found in the new application. There are two types of testing that can be done.

1) manual testing :- Manual testing refers to the process of manually testing software for defects. to ensure that the application's features behave correctly The tester simulates as end-user and tests the application.

2) automation testing :- Automation testing is carried out by running test cases through special automation testing software tools. GUI automation testing aims to reduce

manual testers' efforts so that they only need to write manual testing and automation scripts once.

In CICD, manual testing does not give you better results because it takes time to test the whole application and continuous testing is not possible manually. So GUI automation testing using AI will be useful where the GUI of the application will be tested automatically by the pre-return automation code, which tests the GUI and gives the application report faster compared to manual testing, which reduces the manual efforts of the tester. To automate GUI testing using AI, one needs to write code and test the application multiple times.

1.2 Problem Specification

In traditional software development, life cycle testing of the application is done at the end of the whole application integration. But after the end of the integration, all the functionality of the application is developed. Testing the application is difficult because testing the entire software application takes time. During software application testing, it is necessary to test all of the GUI elements of the software application, verify the functionality of the specific module, check all of the settings, and consider any specific needs of the application. Any single bug in the application could have a significant impact on the software's quality. So, manual testing is time-consuming and not that accurate in terms of testing any big application that has many modules and many functionalities.

There must be accuracy needed at the testing site to test all the GUI elements and the functionality of the application. GUI Automation Testing will solve this by automating the test, which is currently being done manually. By creating the automated scripts, the test gets all the GUI elements tested and provides the results as pass or fail, or any other error that shows the abnormal behaviour of the particular module and that we can easily give to the developer to solve that bug.

1.2.1 Need Analysis

The needs analysis of our study is listed below:

- Demonstrating the technical efficiency of the GUI automation testing approach to accurate testing of the application other than manual testing.
- In one step, with minimal manual effort, you can create a very simple-to-execute

test.

• Examination of the GUI automation testing using an automation framework like Eggplant to automate the testing of the application.

• To provide quality and bug-free applications, GUI automation testing is a very advanced step.

1.3 Scope of Work

To overcome the problem, I came up with a solution implemented in the form of GUI automation testing using AI techniques. To perform this method, eggplant functional, eggplant dai, and eggplant agent are used. This automated test case results in an output that whether the application has any bugs or has any functionality that is not working properly as per the input. During continuous testing

Chapter 2

Literature Survey

In this chapter, I've included a summary of research papers on testing and automation testing, as well as how other researchers did it, what approach they took, and the benefits and drawbacks of those approaches.

2.1 Summary of Related Researches

1) A Metric Framework for the Gamification of Web and Mobile GUI Testing[2, 1]

- **Approach** :- This paper describes the gameplay suggestions and GUI tests. We define "tool and website" as fundamental method concepts, metrics sets, and points. To enable the e-gamified method, system and visual feedback are required. Finally, we have discussed the potential implications. I imagine a road test indicator.

- **Objective** :- The main idea is in our GUI gamification test set. That is a man-made sequence checker in the GUI, which simulates user interaction with AUT (and thus, usually, this is done from the home/main screen).

- **Methodology** :- GUI Gamification metrics based on the test

- **Advantages** :- Modern domain testing is distinguished by the use of a visual user interface (i.e., GUI testing), which focuses on visual interaction with tested applications. The tester should only interact with the system through the user interface during GUI testing. This allows the tester to see the field from the same perspective. The final product is being worked on by the end-user. As the product develops, it is critical to test this functionality. is a website or smartphone app that

displays the feature It is still in its early stages, and many people are contacting it. The GUI is used to create users.

- **Future Scope conclution** :- We outlined the framework for incorporating gamification concepts into GUI test tools in this paper and created conceptual proofs as an existing object plugin. Scout created a test tool to apply concepts to web and mobile applications.

2) How to Create a GUI Map Novel Using a Supported Image Widget detection and segmentation [9, 7]

- **Approach** :- Icons are tested in order to achieve a different GUI component, namely higher accuracy. The Java GUI was used for the majority of previous work. We identified the widgets used in our work. Image processing

- **Objective** :- Consistent software testing This is a costly process that can cost up to 50 percent of the cost of software development and even more on the web, making it critical for security systems. As a result, we will be able to cut costs. Software testing is carried out by converting an existing system into an automatically running system, which is then slowed down. Personal blunder System automation can fail due to errors or omissions, including update failure. Importantly, the automation system has the ability to leave some ambiguity. Changes in assessment quality due to differences in individual abilities

- **Methodology** :- Python accesses various GUI components that use the object-finding algorithm via OpenCV.

- **Advantages** :- It proposes a novel approach to detecting deception. PyAutoGUI automation was used. A wide range of click-through links to a given URL are supported by the Python mouse control library. When you get the screenshot, The KNN section can be used to identify new features and GUI divisions using the OpenCV method. In the event that the outcome is If the screenshot does not match the original displayed GUI, the GUI may appear to be a link or a button. An input box can be identified when the cursor image is activated. It can also be identified as "Composition Box" if a list of items is displayed on the same page.

- **Dataset** :- The training is perfect. The dataset contains both input and response values.

- **Future Scope and Conclusion** :- This allows you to test the accuracy of various real-time websites. We've added features to ten real-time websites. classification and identification accuracy The most important contribution is to reduce staff investment so that we can reduce training and testing time without involving humans. User interaction automation, real-time web application testing, and in-depth user experience are also discussed in this research. In comparison to previous work, it is portable, simplified, and improved. In the future, most real-time scenarios will be used to calculate the accuracy of the GUI acquisition during acquisition, the accuracy of the GUI configuration during partitioning, and the probability of including the test and all analyses.

3) Automatically Generating Test Scripts for GUI Testing[8, 10]

- **Approach** :- We present a standard study aimed at reducing the time it takes to create test documents. A method known as "download and play" records screen actions of user usage and extracts them into text to assist in the development of test documents. One tool that employs this method is Selenium IDE. By the way, compared to directly creating test text by programmes, users who do not have editing skills can create test text using the GUI.

- **Objective** :- To quickly release applications in a short period of time, it is critical to use an automated software test, which requires many hours of development. However, because creating the test documents required for automatic testing takes a long time, the developed teams are too small to be involved in introducing flexible testing tools. We propose a method for checking text in application source code and usable files using direct and flexible analysis in this paper. Experiment results show that the proposed method can reduce the number of human hours by about 61 percent when compared to the traditional method of creating test manuscripts.

- **Methodology** :- test script programs because it is possible to create test documents via GUI.

- **Advantages** :- We were able to reduce the effort by using the method proposed. Approximately 61 percent faster than the standard (productive) method of examining documents using Selenium IDE).

- **Dataset** :- Screen transition

- **Future Scope and Conclusion** :- We have created an automated testing production method. The purpose of the source code and test objectives is to solve a problem that requires time to create test documents. [14, 3]

4) Perfective Pervasive Computing Software Environment: GUI-Oriented Automated Test Platform[6, 5]

- **Approach** :- With comprehensive and continuous applications in computer science development. Software testing [12, 11] is an important phase within the software lifecycle. It is shown to be automatic.Because automated testing can only perform tests, it can save up to 80

- **Objective** :- Graphical User Interfaces (GUIs) are essential natural components of the software for full computers and applications. This paper suggests a solution called GUI Testing Architecture (GUITA) in GUI-focused auto-checking, built on a platform that combines ATS with Winkunner "by socket visual." By default, edit the file and upload it A successful GUI map can also be manipulated to carry out a corresponding test automatically. The solution is guaranteed by an effective and feasible GUI direction method automatic retrieval test, which promotes efficiency and profitability beyond traditional manual testing.

- **Methodology** :- Encapsulating WinRunner in the integrated GUITA framework

- **Dataset** :- The unified framework of GUITA

- **Future Scope and Conclusion** :- We will strengthen and improve the capacity of GUITA next time to gear it up for those who are not good at coding with TSL script. At the same time, due to the incredible WinRunner hole that sees those unfamiliar parts of the Web interface properly, we are going to improve more app methods to solve this problem. In other words, GUITA offers an inexpensive and

possible way to test the GUI and other applications for the company successfully. In addition, this test structure is very suitable for auto-detecting test retrieval targeted at GUIs.

5) Automated GUI Testing for Android News Applications.[4, 13]

- **Approach** :- We want to see if the provided Android phone can display the news app as expected if a good news app is provided. A specific Android phone could be a new type of smartphone, a standard smartphone, a smartphone with new software, or something else entirely. The accuracy and reuse of test terms are important to us. We designed and implemented FLAG, a fully automated mobile GUI testing tool, in this case to perform tests on Oracle reusable without sacrificing test accuracy. For human use, the world automates the process of creating test scenarios and simulates user touch and external screen verification. FLAG began by analysing the GUI structure of a news application and creating test scenarios for all possible GUI functions.

- **Objective** :- FLAG (Automatic Mobile GUI Check) introduces a GUI test tool that aims to make the test oracle reusable without compromising test accuracy. Furthermore, FLAG generates the entire testing process without any human intervention, including production test conditions, user touch simulation, and verification results. Not only are news apps popular, but they also strongly support popular user interactions such as tapping, scrolling, distributing, and sharing. To test flag performance, we chose Android for five commercial phones and five popular news apps in our review.

- **Methodology** :- FLAG overview, test case generator, touch player, The oracle for testing

- **Advantages** :- MSN News App test prediction accuracy

- **Dataset** :- test oracles

- **Future Scope Conclusion** :- VERIFY that the test oracle can be reused without jeopardising the test's accuracy. The procedure begins with a Test Case Generator, which can automatically generate test cases, followed by a Gesture A Player, which

can reproduce all functions in test cases, and finally, a Test Oracle, which can compare the display from a test device with the image from Oracle to determine the test result. Our tests revealed that FLAG had an average accuracy of 95.20 percent, which is significantly higher than 52.6 percent of the time and 5.78 percent for SPAG-C.

Chapter 3

Dataset Description

Manually collected GUI images of the application have been used for GUI automation of the IC-CAP. This dataset has a total of 3000 png images of the GUI elements. These 3000 images are GUI elements of IC-CAP like buttons, menus, icons, and plots, and those images are taken from both the SUT (System Under Test) Windows system and the Linux system. Because the GUI of the application can be changed according to the SUT, the automation scripts for the application dataset are needed to be created according to the test and which GUI elements are required to be tested. The total size of the dataset can go up to 100–500 MB.

3.1 Characteristics of the dataset

- The dataset includes almost all the png images of IC-CAP elements that are needed to test.
- This data set has all the images labeled with their respective elements and in a proper hierarchy, making it easy to code and detect the elements from the screen and test those elements.

Chapter 4

Proposed Method

In this chapter, we discuss which methods I proposed and which automation framework is used to automate the tests for which applications. How can this be implemented and what are the things to be done in which way the proposed work is going to be performed?

4.1 Proposed Research Work

- In the testing for a particular application that is a web-based application or mobile-based, the features of the application must work properly.
- If any functionality of the software or the mobile application is not working properly, then the company can lose a valuable customer. Because the quality of the software application is important for the longevity of the software application, frequent testing is necessary. If the software application has not been tested in recent times, then there is a possibility of getting some bugs. If those bugs are not fixed as soon as possible, the quality of the software application will be compromised.
- So to solve this issue, companies are moving towards the Continuous Integration and Continuous Delivery/Continuous Deployment (CICD) approach, which is different from traditional software development, where the software application is ready to be released at any time during development. Manual testing is not useful for this approach because it is difficult to test frequently with manual testing because it takes time. Automation testing is required to solve this problem. Eggplant is the automation testing framework. It is useful to create an automation test, which was done manually before. Eggplant employs the SenseTalk Language.

- Eggplant components are used in IC-CAP GUI Automation.

  1) Eggplant DAI

  2) Eggplant Agent

  3) Eggplant Functional

  4) System Under Test

4.1.1 Eggplant DAI

- Eggplant Digital Automation Intelligence (DAI) uses a model-based approach to combine automated exploratory testing with directed test automation. The Eggplant AI model is flowchart-based, which shifts focus from coding to the overall user experience.

4.1.2 Eggplant Agent

- The Eggplant Agent creates and connects the Eggplant Function and the Eggplant DAI web application.

4.1.3 Eggplant Functional

- Eggplant Functional is a test automation tool that connects to the system under test (SUT), runs scripts, and enables scripting. Eggplant Functional

4.1.4 System Under Test

- The System which your testing is called System Under Test

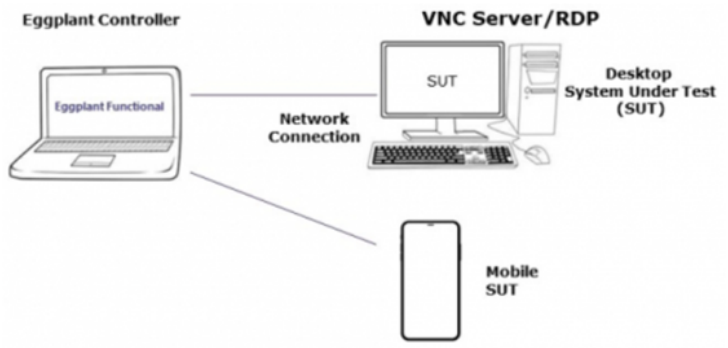

Figure 4.1: **System Overview**

4.2 Workflow

- First, create a manual test case for the system.
- Capture the required GUI images for the test case.
- Write the automation scripts that use the GUI images to automate the test case.
- Then create the model for this test case for better visualization.
- Then attach the script to the model and run it, so every time you need to test this system, there is no need to do it manually. It will be done by this model automatically.

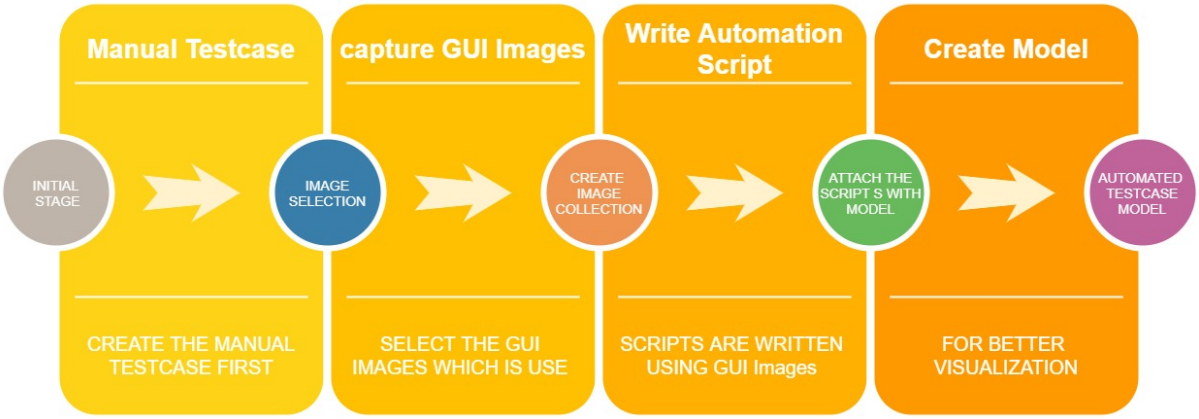

Figure 4.2: **Workflow**

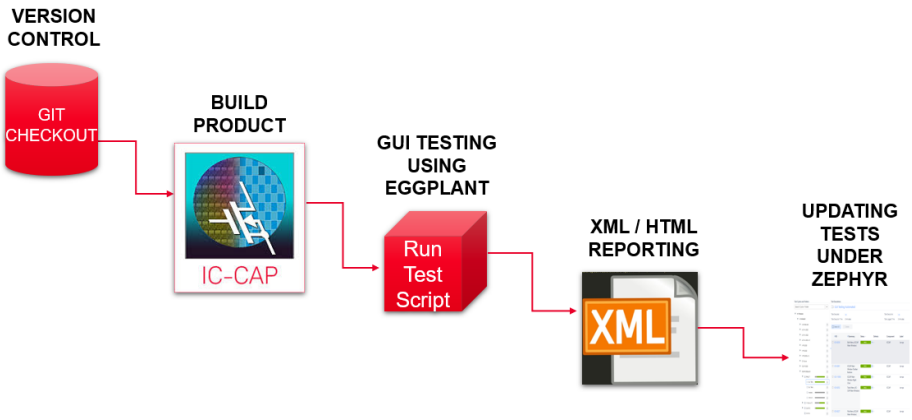

Figure 4.3: **flow of solution**

Chapter 5

Implementation

In this chapter, we discuss the IC-CAP application we are testing on and how much implantation has been done and what the results are of those automated tests.

The industry standard Device Modeling (IC-CAP) software is used for device modelling on the industry standard DC and RF semiconductors. The Analysis Program and Integrated Circuit Characterization (IC-CAP) high speed/digitalis used by extracting accurate compact models, power RF applications, and analogue signals.

5.1 Flowchart Models

- From the current flowchart, we can see that the green dot shows that this feature of the IC-CAP Application is automated and the other is remaining. Then all the actions shown in the flowchart are used in different test cases which test the functionality of that GUI element and check its working.

- We try to replicate the exact look of IC-CAP GUI elements in the Eggplant DAI and add scripts based on their features. The actions are attached to automation scripts.

- All menus, such as the file menu, edit menu, and so on, are linked with their respective menu icons and toolbar icons and are also automated in accordance with the IC-CAP's GUI.

- IC-CAP GUI Automation Model on Eggplant DAI

  1) The IC-CAP Model

2) MODEL OF DATA-VIEWER

5.1.1 The IC-CAP Model

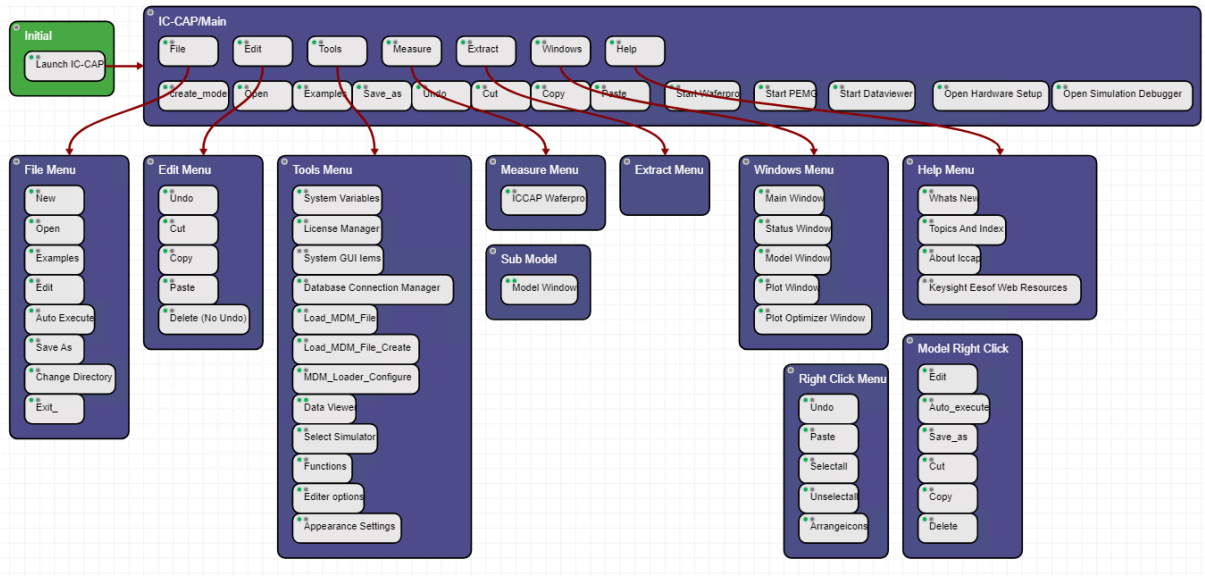

Figure 5.1: **The IC-CAP Model**

5.1.2 MODEL OF DATA-VIEWER

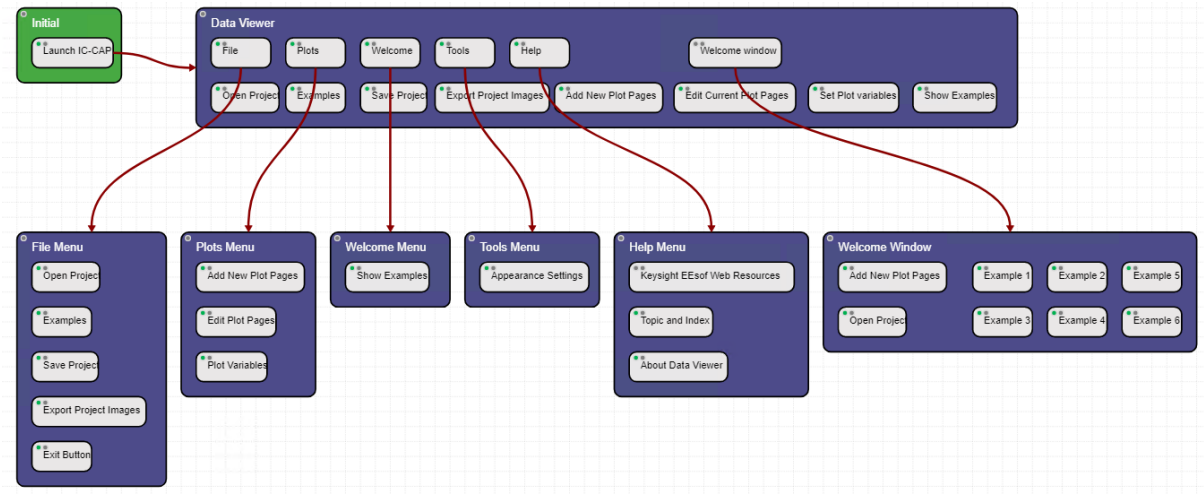

Figure 5.2: **MODEL OF DATA-VIEWER**

5.2 Parallel Execution

- To test the automation scripts on the 3 different SUTs at the same time, we required 3 agents, and all of them had different port numbers. In Drive Mode, Eggplant Functional All SUTs are linked to their own DAI agents.

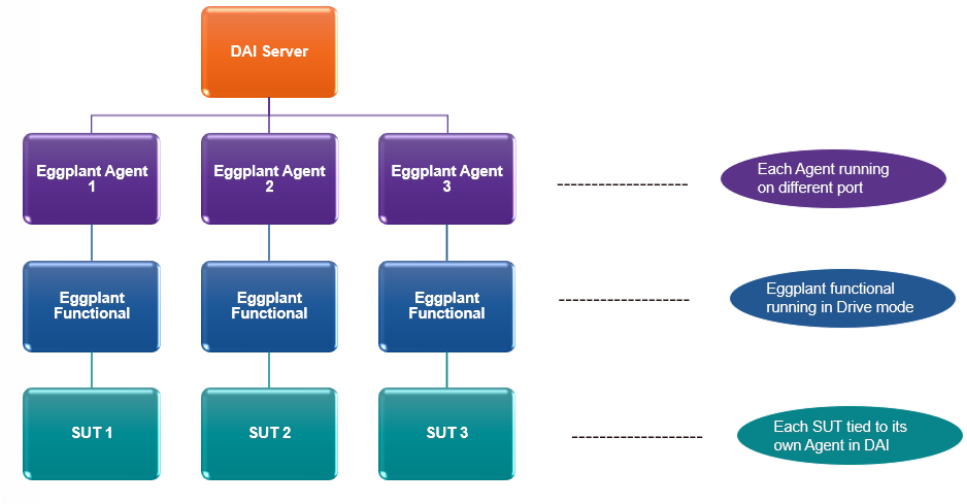

Figure 5.3: **Parallel Execution**

5.3 Previous state vs Current state

- In the previous state, version control and the build product were automated, while the manual testing and manual updating of tests under Zephyr were done manually.

- In its current state, all processes have been automated: first, the version control, then the build product, then the GUI testing using the automation framework eggplant, then the results will be reported to XML/HTML reporting, and finally, the file updating tests under the zephyr will show the results of all tests.

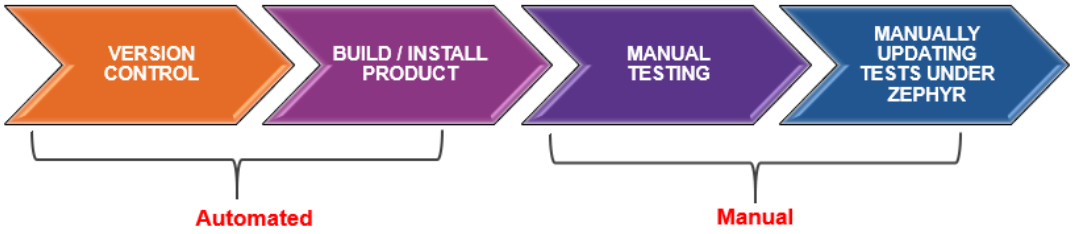

Figure 5.4: **Previous state**

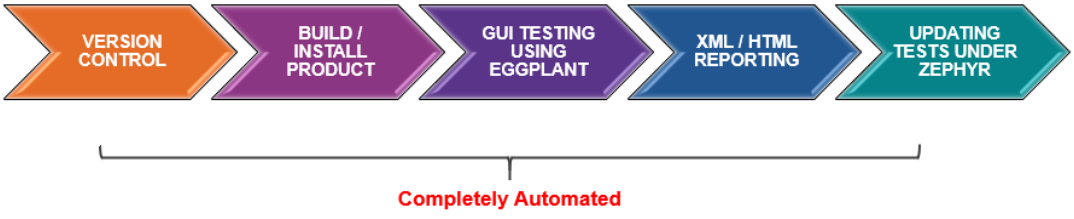

Figure 5.5: **Current state**

5.4 The Benefits of CI/CD with Eggplant

- The ability to run parallel tests on multiple platforms at the same time.
- Earlier detection of defects.
- More Accurate Tests and Eliminating Human Errors
- Early Access and minor update releases
- Better Utilization of Manpower and Reduced Business Costs.
- A shorter software development cycle and more frequent releases.

5.5 CI/CD Jenkins Pipeline

In the CI/CD Jenkins Pipeline, the execution starts with the GIT checkout, taking the latest code, and installing the latest build. Then the eggplant will run the automation script and the end result will be passed to Zephyr.

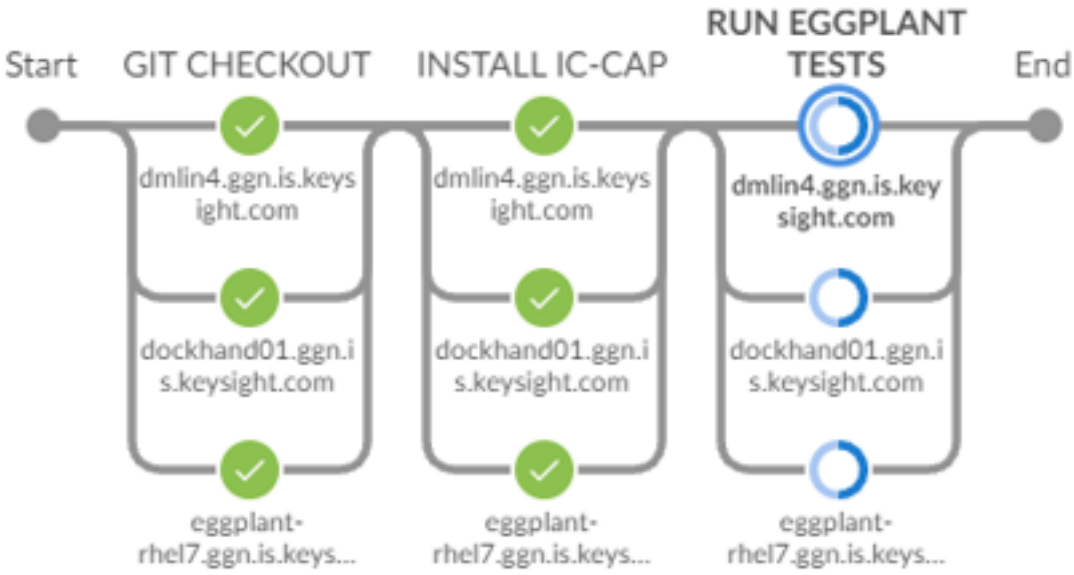

Figure 5.6: **CI/CD Jenkins Pipeline**

Chapter 6

Results

For IC-CAP, the total manual testing hours required is 460 hours (all platforms), and 170 hours can be automated using Eggplant. We are automating the 170 hours of manual testing into automation testing by implementing automation testing with the AI automation framework Eggplant. After automating the test, we now require only 8 hours. which required 170 hours previously. So it's almost 20x–21x faster than manual testing. We are testing IC-CAP on three systems: one is Windows, and the other two systems are Linux systems (Rhel7 and Suse12).

6.1 Results of the Automated Tests

Type of Testing	**Hours**
Complete Manual Testing	**460**
Manual Testing(which can be Automated using Eggplant)	**170**
Eggplant Automation Testing	**8**

Table 6.1: **Results of the Automated Tests**

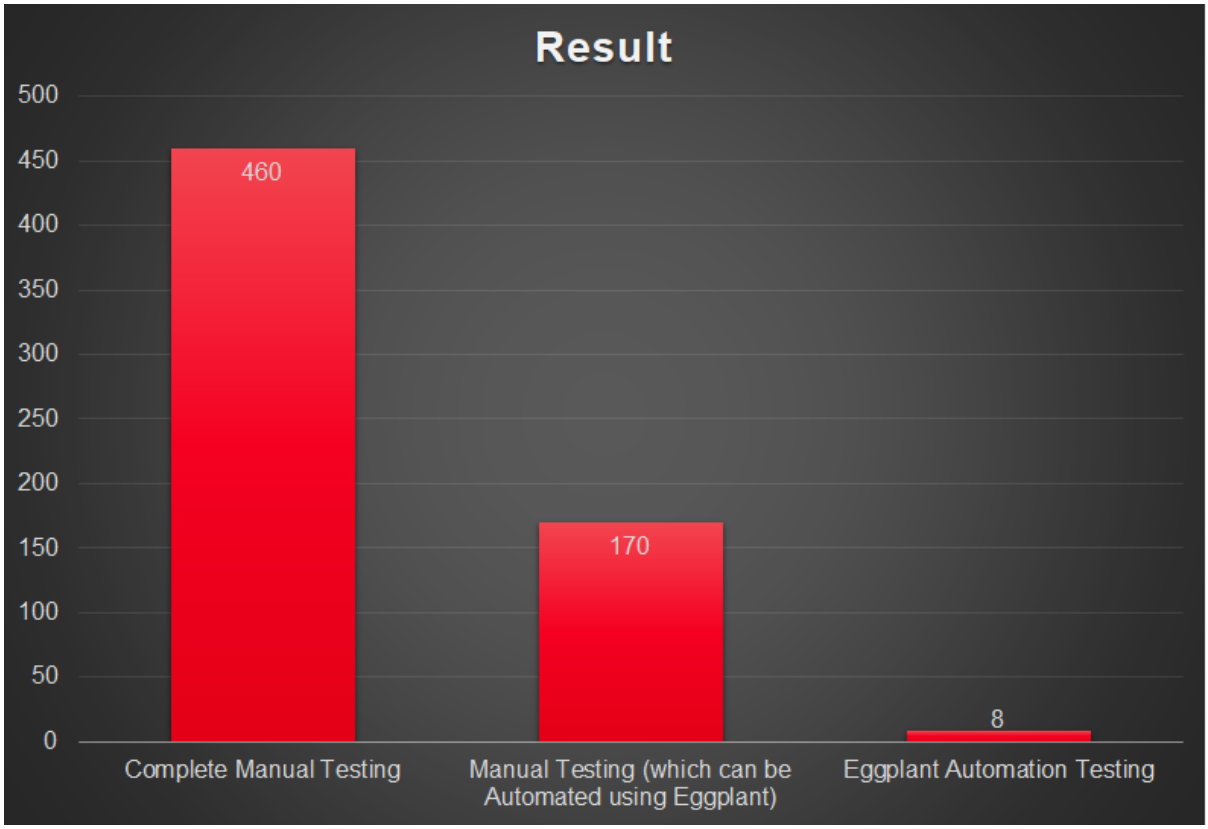

Figure 6.1: **Results of the Automated Tests**

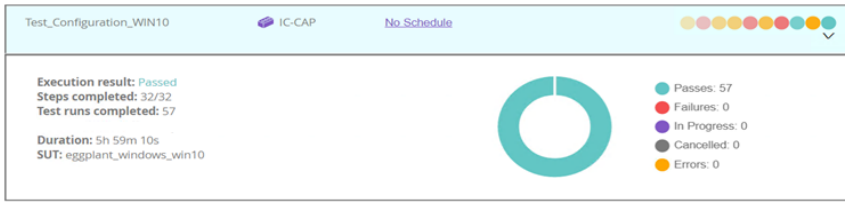

Figure 6.2: **Win10 Result**

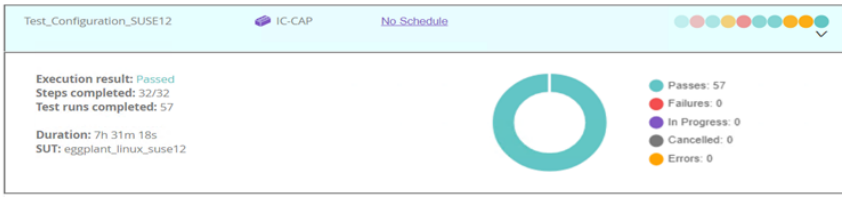

Figure 6.3: **Suse12 Result**

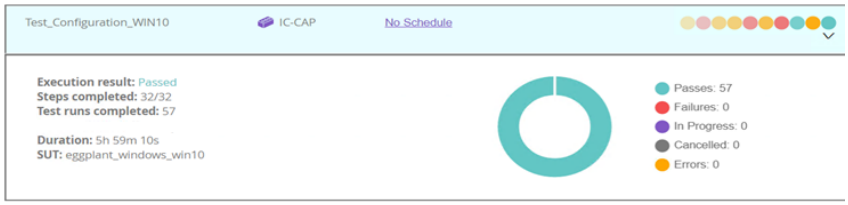

Figure 6.4: **Win10 Result**

Chapter 7

Limitations and Future Work

The eggplant tool is best suited for faster GUI tests or test operations. Eggplant DAI "expects" to hear from the agent periodically. If it hasn't gotten any data from the agent for 2 minutes, it assumes an issue and times out. Waiting time is not recommended for eggplant.

Due to the time-out limitation of Eggplant, the squads need to use a creative approach by creating GUI test cases that don't involve idle times (simulation, optimization, etc.) of more than 2 min. Focus on GUI usage, not time-consuming simulations.

IC-CAP includes a lot of tests that are performed manually and cannot be automated, like instrument testing, antivirus, surveys, fresh machines, remote display, etc. Some long-running tests are better suited for application-level automation.

7.1 Future Work

- Automate all possible tests that can be automated in IC-CAP GUI Testing.
- Eggplant Integration with Jenkins

  Running Eggplant tests from Jenkins and reporting results
- Zephyr for JIRA.

  Zephyr allows you to create, manage, execute, and report tests.
- Zephyr Integration

  Reporting eggplant results to Zephyr

Chapter 8

Conclusion

In this report, I talked about a GUI automation testing framework based on Eggplant to address the issue of manual testing in Agile Models. It takes time to test the software application. Meanwhile, to solve that issue, I implemented this **GUI Automation Testing** and verified the effectiveness and robustness of the automation testing. Furthermore, the one-click solution for testing the software application fully automates the process. With the comparative study of the automation framework, my final outcome is test results that are **20x–21x faster** than manual testing.

Bibliography

[1] Young-Min Baek and Doo-Hwan Bae. Automated model-based android gui testing using multi-level gui comparison criteria. In *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 238–249, 2016.

[2] Filippo Cacciotto, Tommaso Fulcini, Riccardo Coppola, and Luca Ardito. A metric framework for the gamification of web and mobile gui testing. In *2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 126–129, 2021.

[3] Jin Chen, Mengxiang Lin, Kai Yu, and Bing Shao. When a gui regression test failed, what should be blamed? In *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*, pages 467–470, 2012.

[4] Wontae Choi, Koushik Sen, George Necul, and Wenyu Wang. Detreduce: Minimizing android gui test suites for regression testing. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, pages 445–455, 2018.

[5] Edward T.-H. Chu and Jun-Yan Lin. Automated gui testing for android news applications. In *2018 International Symposium on Computer, Consumer and Control (IS3C)*, pages 14–17, 2018.

[6] Riccardo Coppola, Maurizio Morisio, and Marco Torchiano. Maintenance of android widget-based gui testing: A taxonomy of test case modification causes. In *2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 151–158, 2018.

[7] Gennaro Imparato. A combined technique of gui ripping and input perturbation testing for android apps. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 2, pages 760–762, 2015.

[8] Muneyoshi Iyama, Hiroyuki Kirinuki, Haruto Tanno, and Toshiyuki Kurabayashi. Automatically generating test scripts for gui testing. In *2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 146–150, 2018.

[9] K. Jaganeshwari and S. Djodilatchoumy. A novel approach of gui mapping with image based widget detection and classification. In *2021 2nd International Conference on Intelligent Engineering and Management (ICIEM)*, pages 342–346, 2021.

[10] Ying-Dar Lin, Edward T.-H. Chu, Shang-Che Yu, and Yuan-Cheng Lai. Improving the accuracy of automated gui testing for embedded systems. *IEEE Software*, 31(1):39–45, 2014.

[11] Scott McMaster and Atif M. Memon. An extensible heuristic-based framework for gui test case maintenance. In *2009 International Conference on Software Testing, Verification, and Validation Workshops*, pages 251–254, 2009.

[12] Tiago Monteiro and Ana C.R. Paiva. Pattern based gui testing modeling environment. In *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops*, pages 140–143, 2013.

[13] Abdul Rauf and Mohammad N. Alanazi. Using artificial intelligence to automatically test gui. In *2014 9th International Conference on Computer Science Education*, pages 3–5, 2014.

[14] Yepeng Yao and Xuren Wang. A distributed, cross-platform automation testing framework for gui-driven applications. In *Proceedings of 2012 2nd International Conference on Computer Science and Network Technology*, pages 723–726, 2012.

Appendix A

Eggplant functional Learning

`https://www.eggplantsoftware.com/eggplant-functional-expert`

`https://www.eggplantsoftware.com/eggplant-functional-level-2-genius`

`https://www.eggplantsoftware.com/eggplant-ai-training-and-certification` Eggplant Functional Documentation

`https://docs.eggplantsoftware.com/ePF/eggplant-functional-documentation-home.html`