Grenze International Journal of Engineering and Technology, June Issue

GUI Testing using AI Automation Framework

Dipendra Dabhi[1], Vijay Ukani[2] and Sandeep Mehta[3]
[1-2]Department of CSE, Institute of Technology, Nirma University Institute of Technology, Ahmedabad, India
Email: dabhidipendra06@gmail.com, vijay.ukani@nirmauni.ac.in
[3]Department KTO PSS, Keysight International Private Limited, Gurgaon, India
Email: sandeep mehta@keysight.com

***Abstract*—In the waterfall model of software engineering, all the testing that is performed on the software application is done at the end of the software development life cycle, and the testingis done manually. It is very time-consuming and not that feasible to meet the customer's needs. For the best approach to developingsoftware, we are shifting to the agile model, which is based on theContinuous Delivery/Continuous Deployment (CI/CD) approach.In an agile model, we repeat all of the software development life cycle processes multiple times, very frequently, before delivering the software to the customer. In order to get feedback and processfurther, in this scenario, the testing of the software application will be done frequently as the software build needs to be released frequently. And the testing of software manually takes time. Howto resolve this problem? Automation testing is used when newly released software is tested automatically using an automationframework and the result is passed if a bug is found. It saves so much time for the quality assurance team in testing software applications multiple times and infrequent releases.**

***Index Terms*— Automation Testing, GUI Testing, CI/CD, Agile.**

I. INTRODUCTION

In software development currently, we are using the waterfall model. In this model, testing is done at the end of the whole software development life cycle. It is very time-consuming if we find any bug in the software, as it needs to pass through all the previous steps. And in this faster development. A single bug can also change the major consequences of the application. If the single functionality of an application is not working, then the user can try another software. To engage the user with the software application, the bug and its features need to be fixed quickly. And give the user better quality assurance. To solve this problem, the Agile Model is used. Which employs continuous integration. To allow for more rapid deployment of the software application's new features, 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**:-The process of manually testing software for defects is called "manual testing."
   to ensure the correct behavior of the application features.The tester plays the role of an end-user and tests the application.
2) **Automation Testing**:-Automation testing is done by using special automation testing software tools to perform and execute the test cases suite.

The goal of GUI automation testing is to reduce the number of manual testers' efforts so that they only need to write manual testing and automation scripts once.

Grenze ID: 01.GIJET.8.2.42_1

In CI/CD, manual testing does not give you better results because it takes time to test the whole application and continuous testing is not possible manually.
As a result, automation testing with the Automation Framework will be beneficial, as the application's GUI 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 testing using the Automation Framework, we can write the code once and test the software application multiple times.
Your goal is to simulate the usual appearance of papers in Conference Proceedings or Journal Publications. We are requesting that you follow these guidelines as closely as possible.

II. Related Work

In recent years, there have been numerous attempts have been done in the field of testing. and especially in the GUI (graphical user interface) testing. [2]
A researcher proposes gamification for exploratory GUI (graphical user interface) testing. To enable a gamification approach for the tester, they used visual feedback, a scoring scheme, and a set of metrics. In the end, they concluded that they had defined the framework of the gamification concept in GUI (graphical user interface) testing tools. [1] They implemented various plugins like "name scout" and "exploratory testing tools" to apply the concepts of mobile and web applications.
Another researcher's proposal for GUI Map Novel uses supported image widget detection and segmentation. The approach to this is to test icons to achieve a segmented GUI segment for better accuracy. Their previous work was mostly done using Java GUI, where they discovered the widget types in their work. [4], [5]
The methodology they used was Python and OpenCV to get different GUI components that are used in object detection, and the dataset they used was training optimal data. They concluded that the main contribution was reduced when including input data as well as response values. Training and testing time is reduced without human intervention. [6], [8]
Another researcher's proposal for automatically generating test scripts for GUI testing where used the method called "download and play again" which records the user interaction with the screen action and extracts it with text. They used the Selenium IDE tool- for this approach. Their objective is to quickly release software applications during the frequent release cycle. [3], [10]
Automation testing is required. And reduce the number of human testing hours. The methodology is used by test script programmers because it is possible to create the scripts through the GUI. A dataset is used for the screen transition. In the end, they concluded that a production method for automatic testing was needed. Source code and testing objectives to solve the problem that it takes time to create test documents. [7], [9]

III. Problem Specification

Testing the application is very hard because it takes time to test the whole software application. It needs to test all the GUI elements of the software application, verify the functionality of the particular module, and check all the settings. Any specific needs of the application must be considered at the time of testing the software application. Any single bug in the application may possibly have a major impact on the quality of the software. Currently, we are using the waterfall model. Which is very time-consuming. 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.
Your goal is to simulate the usual appearance of papers in Conference Proceedings or Journal Publications. We are requesting that you follow these guidelines as closely as possible.

IV. Proposed Research Work

Businesses are shifting to the Agile Software Development model, which employs a CI/CD approach, as opposed to the traditional waterfall model for software development, in which the software application is ready for release at any time during development. For this approach, manual testing is not that useful because, with manual testing, it is not possible to test frequently because manual testing takes time. To solve this problem, automation testing is required. Eggplant is the AI automation testing framework. It is useful to create an automation test, which was done manually before. Eggplant employs the Sense Talk language.

A. Eggplant's artificial intelligence

In Eggplant, AI is used by monitoring the user's interactions with the technology, and how the interactions will respond to that action will be monitored.

This allows testing the whole user experience, including all the usability and performance, across all operating systems, browsers, or devices.

The AI-driven Eggplant test method creates a system model and a user journey and automatically generates test cases that provide a solid integration of the user experience with system performance and functionality. By using automated response loops, testers can quickly identify and address issues to ensure their product continues to please users and aligns with user objectives. And because QA makes sure that any UI errors, bugs, or operational issues are identified and fixed before they are released, QA can remove the negative brand image and user impact often associated with technical errors.

B. Eggplant components

Eggplant components are used in IC-CAP GUI Automation.

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.
2) Eggplant Agent:
   • Eggplant Agent creates and connects the Eggplant Function and the Eggplant DAI web application.
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) System Under Test:
   • The system that you are testing is called the "System Under Test."

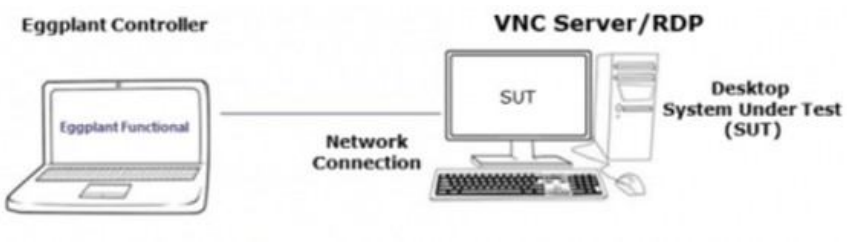

Fig. 1. System Overview

C. 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 it creates a 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.

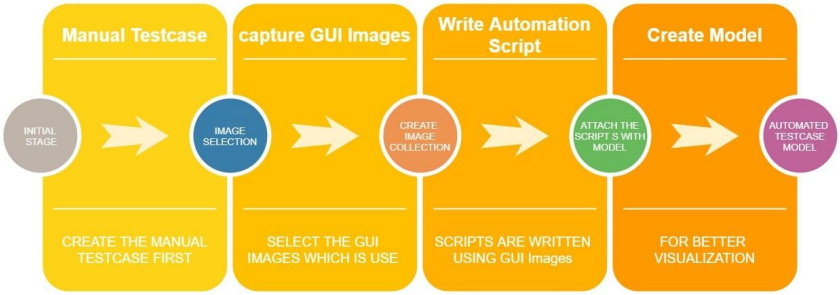

Fig. 2. Workflow

V. IMPLEMENTATION

For GUI Automation Testing, here we automate the test for the IC-CAP. The industry standard Device Modeling (IC-CAP) software is used for device modeling on industry-standard DC and RF semiconductors.
The Analysis Program and Integrated Circuit Characterization (IC-CAP) high speed/digitalis are used by extracting ac- curate compact models, power RF applications, and analog signals. In the previous case, testing on IC-CAP was done manually. Now we are testing the IC-CAP using automation testing.
Zephyr is a test management tool. It creates, plans, and executes manual tests. is tracking manual testing for the project. Each Zephyr test can have detailed steps. A test cycle is a focused set of test cases that are grouped to achieve specific testing goals.

A. Previous state vs Current state

- In the previous state, version control and the building 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 building 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

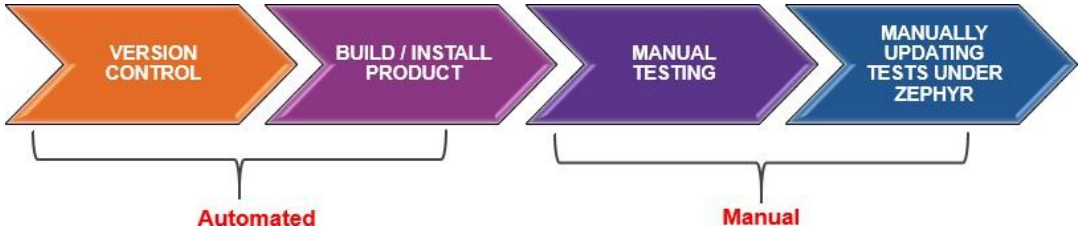

Fig. 3. Previous state

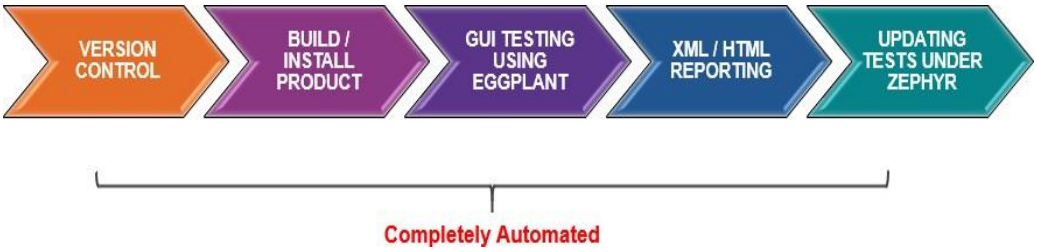

Fig. 4. Current state

B. 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.

C. 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.

VI. RESULTS OF THE AUTOMATED TESTS

For IC-CAP, the total manual testing hours required is 170H (all platforms). 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).

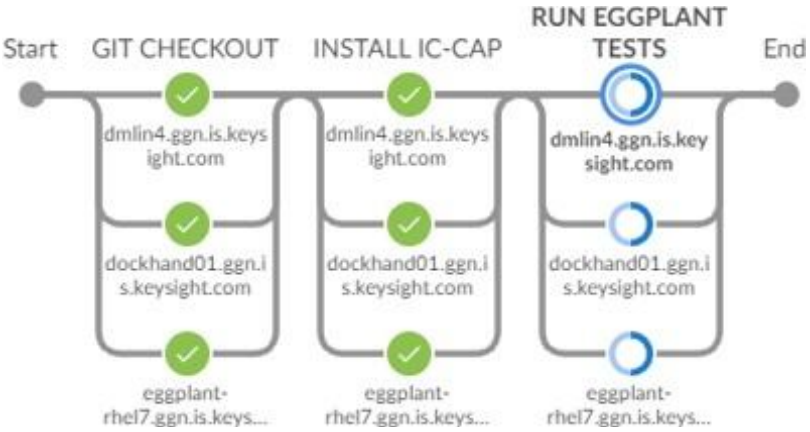

Fig. 5. CI/CD Jenkins Pipeline

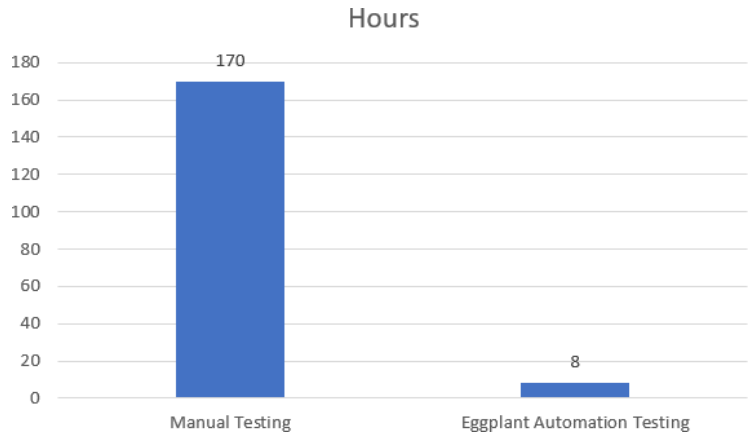

Fig. 6. Results of the Automated Tests

TABLE I: RESULTS OF THE AUTOMATED TESTS

Type of Testing	Hours
Manual Testing	170
Eggplant Automation Testing	8

VII. CONCLUSIONS

In this paper, 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.

REFERENCES

[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 Engineer- ing (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] 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.

[4] 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.
[5] Riccardo Coppola, Maurizio Morisio, and Marco Torchiano. Mainte- nance 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.
[6] 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.
[7] 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.
[8] 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.
[9] 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.
[10] 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.