International Journal of Computer Trends and Technology
ISSN: 2231–2803 / https://doi.org/10.14445/22312803/IJCTT-V72I9P118

Volume 72 Issue 9, 120-125, September 2024
© 2024 Seventh Sense Research Group®

Review Article

AI Agentic Scriptless Automation in Software Testing

Ganesh Viswanathan

3530 Alister Ave SW, Concord, NC, USA.

Corresponding Author : cool.ganesh.v@gmail.com

Received: 01 August 2024 Revised: 31 August 2024 Accepted: 23 September 2024 Published: 30 September 2024

Abstract *- Software testing is a critical phase in the software development lifecycle, yet traditional test automation approaches remain time-consuming, resource-intensive, and reliant on deep technical expertise. Script-based automation frameworks demand continuous maintenance as application changes occur, often leading to inefficiencies and flakiness in modern agile development environments. In response, scriptless automation has transformed software testing by introducing high-level interfaces that simplify the automation process. Unlike conventional automation, which depends heavily on scripting, scriptless automation offers a more accessible, maintainable approach through visual modeling, keyword-driven testing, and data abstraction. This paper provides a comprehensive technical overview of scriptless automation, detailing its architecture, core components, and integration with CI/CD pipelines. It highlights the benefits of scriptless automation, such as reduced complexity, ease of maintenance, and faster adaptation to changing software environments, while also addressing its limitations. The discussion then moves to the next evolutionary step in software testing: leveraging autonomous AI agents to manage the creation, execution, and maintenance of test cases without human intervention. This AI-driven approach offers significant advantages, including reduced testing cycles, enhanced test coverage, accelerated time to market, and improved collaboration between technical and non-technical teams. In addition, the paper explores real-world implementations of AI-driven scriptless automation, examining both the merits and challenges of deploying this cutting-edge technology in diverse software environments. The insights presented will enable organizations to optimize their testing strategies and improve the quality and speed of software delivery in an increasingly complex digital landscape.*

Keywords *- Continuous integration, Keyword-driven testing, Scriptless automation, Software testing, Test automation, AI agents, Autonomous testing, Application under test, Continuous deployment.*

1. Introduction

The rapid evolution of software development methodologies, particularly Agile and DevOps, has placed unprecedented demands on testing processes. Continuous integration and delivery (CI/CD) pipelines require fast, reliable, and scalable test automation. However, traditional test automation is often ill-suited to meet these demands due to its reliance on complex scripting, high maintenance overhead, and the need for specialized technical skills.

As applications evolve, the frequent updates and modifications required to maintain test scripts become both time-consuming and error-prone, often causing automation to become a bottleneck rather than an enabler of agility. This challenge is further exacerbated by the fact that traditional automation frameworks are rigid and unable to keep pace with the dynamic nature of modern software development. Each code change may require manual updates to the test scripts, introducing delays, inefficiencies, and increased costs. Moreover, the high technical expertise required to create and maintain scripts limits the involvement of non-technical team members, creating a dependency on specialized testers. This creates a significant research gap: the need for automation tools that can adapt to changing application landscapes without manual intervention while being accessible to a broader range of users. Scriptless automation has emerged as a solution to these challenges by eliminating the need for detailed scripting. Using visual modeling, keyword-driven frameworks, and data abstraction, scriptless tools simplify the creation and maintenance of automated test cases. This enables faster test case development, reduces maintenance overhead, and allows for broader team involvement, including business analysts and non-technical stakeholders. However, despite its advantages, scriptless automation still requires human oversight to execute, maintain, and adapt test cases, particularly in response to application changes. To address this problem, the next evolution in testing is AI-driven agentic scriptless automation. This approach integrates autonomous AI agents with scriptless frameworks, allowing the automation to become self-sustaining, adaptive, and intelligent. AI agents can dynamically generate, execute, and maintain test cases without the need for constant human intervention, enabling true continuous testing. By leveraging machine learning and other AI techniques, these agents can respond to changes in the application, self-heal failing test cases, and ensure comprehensive test coverage. This evolution promises to bridge the gap between the limitations of traditional and scriptless automation, enhancing efficiency, improving test accuracy, and accelerating the overall development process. This paper addresses the research gap by exploring the technical underpinnings of scriptless automation and introducing the concept of autonomous AI-driven testing agents. It delves into the architecture, operational mechanics, and integration of this approach within modern CI/CD environments, offering a glimpse into the future of truly intelligent, self-sustaining test automation.

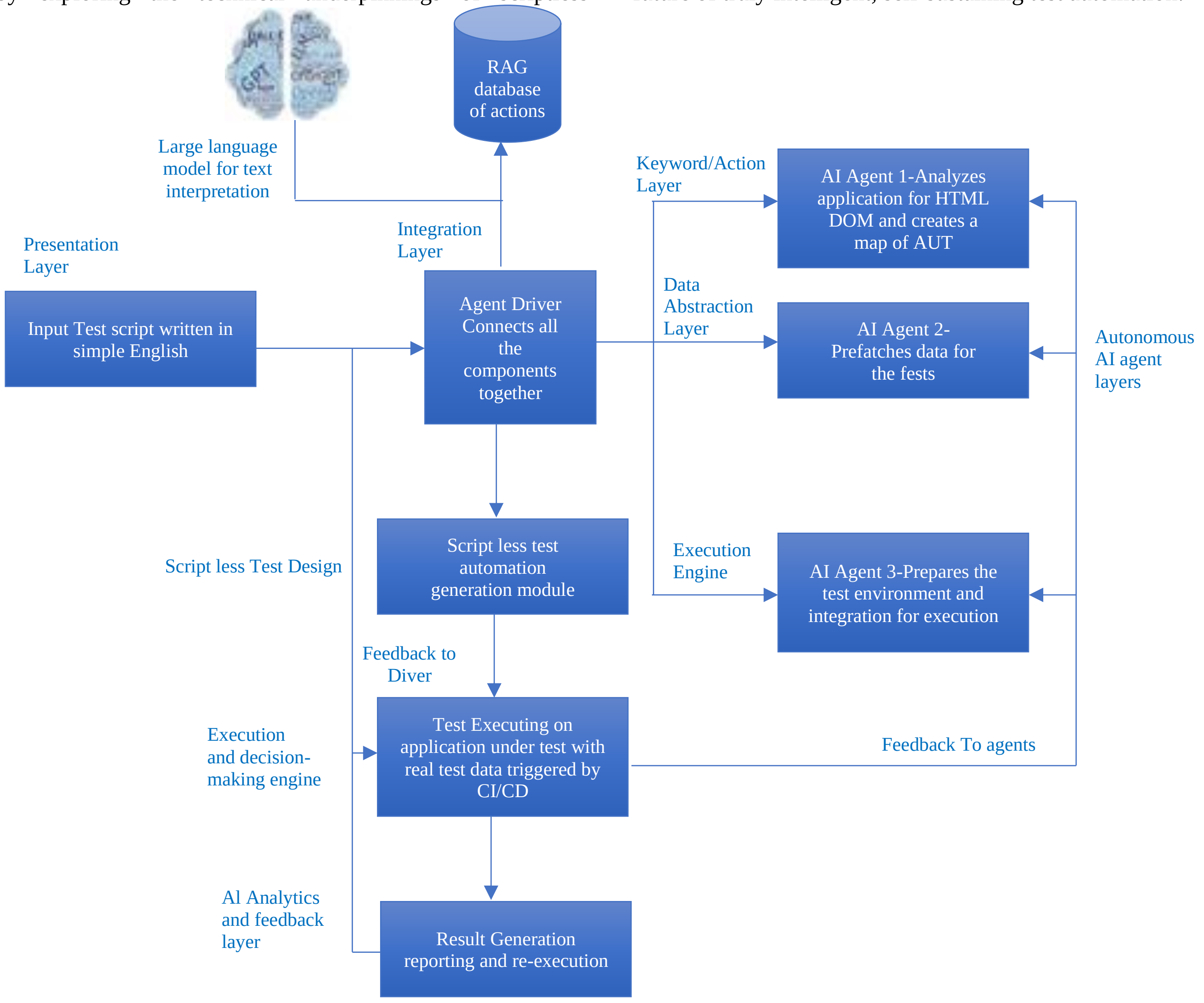
RAG database of actions
Large language model for text interpretation
Keyword/Action Layer
AI Agent 1-Analyzes application for HTML DOM and creates a map of AUT
Integration Layer
Presentation Layer
Input Test script written in simple English
Agent Driver Connects all the components together
Data Abstraction Layer
AI Agent 2-Prefatches data for the fests
Autonomous AI agent layers
Script less Test Design
Script less test automation generation module
Execution Engine
AI Agent 3-Prepares the test environment and integration for execution
Feedback to Diver
Execution and decision-making engine
Test Executing on application under test with real test data triggered by CI/CD
Feedback To agents
AI Analytics and feedback layer
Result Generation reporting and re-execution

Fig. 1 Architecture of scriptless automation tools

2. Scriptless Automation: Technical Overview

2.1. Architecture of Scriptless Automation Tools

At the core of scriptless automation tools is a multi-layered architecture designed to decouple test logic from the underlying application code. This architecture typically includes:

- *Presentation Layer*: A User Interface (UI) that allows users to create and manage tests via drag-and-drop operations, form-based inputs, or Natural Language Processing (NLP).
- *Keyword/Action Layer*: A repository of pre-defined keywords or action words that represent common test operations (e.g., "Click," "Input Text," "Verify Element"). These keywords are often parameterized to handle dynamic data inputs.
- *Execution Engine:* The component responsible for interpreting the high-level test instructions generated by the user and converting them into executable actions against the Application Under Test (AUT).
- *Integration Layer:* Interfaces for connecting with CI/CD pipelines, test management systems, and version control systems.
- *Data Abstraction Layer*: Handles test data management, allowing users to define data-driven tests without embedding data directly in test cases.

2.2. Architecture of Agentic Scriptless Automation Tools

AI agent-driven scriptless automation systems are built on a multi-layered architecture, where autonomous AI agents operate within a traditional scriptless automation framework enhanced by AI capabilities. The additional components of this architecture include**:**

- *Autonomous AI Agent Layer*: AI agents are the core of this architecture. These agents are powered by advanced machine learning models that enable them to understand, adapt, and optimize test processes. The agents can automatically generate, execute, and maintain test cases, learning from past experiences to improve future testing cycles. They can make decisions independently, such as selecting the best test strategies or identifying the most critical areas of the application under test.

- *Execution and Decision-making Engine:* The execution engine is driven by AI agents that translate high-level testing objectives into executable actions. This engine interacts with various automation tools and frameworks, executing tests across different environments. The AI agents continuously monitor test execution, making real-time decisions to optimize test coverage and execution efficiency.

- *AI Analytics and Feedback Layer:* This layer utilizes AI to analyze test data and generate insights. AI agents continuously learn from the results of each testing cycle, adapting future tests to improve coverage and accuracy. This feedback loop ensures that the testing process becomes more intelligent and efficient over time.

3. Key Technical Components

3.1. Visual Test Creation Interface

The visual interface of a scriptless automation tool is typically designed as a web-based or desktop application that offers drag-and-drop functionality. This interface often includes advanced features like test case modeling, test data binding, and error handling. Modern web frameworks facilitate the development of scalable interfaces that seamlessly integrate all underlying functions. These tools are designed with drag-and-drop simplicity for basic users while also providing IDEs for those requiring more advanced capabilities. By abstracting the complexities, the interface allows users to concentrate on defining the "what," leaving the "how" to be efficiently managed by the underlying automation engine.

3.2. Keyword-Driven Framework

Keywords are reusable methods or functions that encapsulate specific test actions. They are implemented as a combination of scripts and libraries within the tool. For example, a "Click Button" keyword might locate the button element and execute a click event using a browser driver. The elements outlined above represent the core components of any UI web application. Each of these entities is associated with a defined set of actions, such as click, double-click, drag-and-drop, set, delete, clear, submit, hover, activate, deactivate, and more. By establishing interfaces for these actions, it is possible to create a comprehensive keyword library that supports automation across various web applications. In UI testing, the process typically involves executing a sequence of actions on these entities (test steps) and then validating the application's state after those steps. With the defined keywords, action methods can be used to carry out test steps, while entity state methods determine the final states of UI elements. These states can then be asserted and validated against expected results to ensure the application under test is functioning correctly.

Windows	Menus	Controls
• Main window • Child windows • Pop-up windows • Dialog windows	• Menu bars • Drop-down menu • Context-aware menu	•Buttons • Text boxes • Links • Radio buttons • Checkboxes • Drop-down select boxes • Sliders • Tabs • Scroll bars

Fig. 2 Key technical components

3.3. Test Execution and Reporting

The execution engine processes keyword-driven instructions and translates them into actionable steps. During this process, it interacts with the Application Under Test (AUT) using browser drivers, mobile testing tools, or API testing tools. Several engines available today can perform this conversion effectively.

They can be divided into 3 major categories:

3.3.1. HTML-DOM Driven Engines

Traditionally, HTML DOM manipulation has been the foundation for interacting with web application elements. Early automation tools used browser-specific DLLs to create handles for these elements, controlling their behavior through object methods. However, this approach was OS and browser-dependent, often requiring extensive code for even simple operations and lacking compatibility. The introduction of web drivers marked a significant leap forward. These drivers provided standardized interfaces for web elements, facilitating code sharing and enabling the automation of any application adhering to W3C standards. This led to the development of open-source libraries and frameworks like Selenium, Appium, Winium, Watir, Cypress, Playwright, Puppeteer, Robot Framework, TestCafe, WebDriverIO, Nightwatch.js, and Katalon Studio. These advancements propelled UI test automation, and many scriptless tools began leveraging these technologies. Even modern RPA tools like Automation Anywhere, UiPath, Robocorp, and Blue Prism rely heavily on these UI automation frameworks. Despite improvements, UI tests remain fragile, as changes to the DOM during deployments can render test suites obsolete, requiring constant maintenance. Data management also poses a challenge, as UI tests often interact with integrated systems, making it difficult to maintain a consistent test data set. Artificial intelligence offers a solution by optimizing and self-healing tests, dynamically identifying the best XPaths for elements during test execution. Scriptless automation tools abstract these DOM complexities, providing simple wrapper functions that reduce boilerplate code and expedite test automation.

3.3.2. Image Recognition Engines

Image recognition engines simulate user interactions by recognizing on-screen elements rather than working directly with the HTML DOM. Historically, these engines were slower and more error-prone due to their dependence on screen resolutions and were often used as fallback mechanisms to DOM-driven engines. Tools like Applitools have challenged this perception, establishing a market for image recognition engines. Libraries such as Tesseract, AutoIT, and Sikuli have further advanced this field. With the rise of large image models like DALL-E, significant improvements in recognition and the ability to respond dynamically to changes in applications are expected. Scriptless automation tools now include libraries that support image recognition tasks, offering a rich set of methods for UI automation through visual cues.

3.3.3. API and Database Engines

With the advent of services, particularly microservices, much of today’s business logic resides in middleware. Traditionally, backend testing involved direct database interactions using DB drivers, SQL queries, and assertions. While this approach is still relevant, API testing has become more prevalent. Libraries like Rest Assured and Requests, along with tools like Postman, facilitate the testing of middleware business logic. Scriptless automation tools provide wrapper methods for the most common API and database testing functions across a varied set of types and platforms, streamlining the testing process across different layers of an application.

Test results are captured in real-time and are visualized through dashboards or exported into reports. All scriptless automation tools offer customizable dashboards and API integrations, allowing data to be pulled into other enterprise repositories seamlessly again; this is another layer where AI, specifically Gen AI, can help make sense of these reports and provide actionable feedback to developers.

3.4. Autonomous Test Case Generation using AI Agent

AI agents are capable of autonomously generating test cases by analyzing the AUT and user interaction patterns. This capability reduces the need for manual test case creation and ensures comprehensive test coverage that adapts to changes in the application.

3.5. Adaptive and Self-Optimizing Agentic Testing

AI agents enable adaptive testing by dynamically adjusting test cases and execution strategies based on real-time data. These agents continuously optimize testing processes, ensuring that tests remain relevant and effective as the software evolves.

3.6. Real-Time Analytics and Predictive Feedback

AI agents provide real-time analytics, offering insights into test performance, defect trends, and overall system stability. Predictive analytics allow these agents to forecast potential issues, enabling preemptive actions to be taken, thus reducing the likelihood of critical defects reaching production.

4. Industry Leaders in AI-Driven Scriptless Automation Today

Scriptless user interface automation is not a new concept in the industry; it has been around for decades, beginning with pioneers like Rational Robo, WinRunner, and Quick Test Professional in the 1990s. These early tools were based on the "record and playback" approach, where users manually guided the tool to record interactions with the application.

However, recent advancements in AI and Generative AI have significantly enhanced these tools, automating much of the manual guidance that was once required. Today, several leading tools are making strides in this space:

4.1. Testim

Testim uses AI to help automate the creation and maintenance of test cases. It offers a scriptless interface where users can create tests by interacting with the application, and the AI engine helps in identifying, managing, and updating elements in the application as it evolves. It also involves a self-healing feature that is useful when application code changes with deployments.

4.2. Test.ai

Test.ai focuses on mobile and web application testing, using AI to generate and maintain test cases automatically. It is particularly strong in understanding user interfaces and adapting to changes.

4.3. Katalon Studio

Katalon Studio provides both script-based and scriptless options; its AI-driven features make it a strong contender in the scriptless testing space. It allows testers to create automated tests through a visual interface, with AI assisting in test creation and maintenance.

4.4. Leapwork

Leapwork is a no-code automation platform that allows for creating test cases visually. It uses AI to assist in test creation and handle dynamic elements in the UI, making it easier to maintain tests.

4.5. Eggplant
Eggplant provides a model-based approach to test automation, using AI to create and maintain tests. It supports a wide range of platforms and applications, and its AI-driven approach helps in reducing the maintenance overhead.

4.6. ACCELQ
ACCELQ is a cloud-based test automation tool that focuses on AI-driven, scriptless testing. It offers a codeless interface for test creation and uses AI to maintain tests as applications evolve.

4.7. Applitools
Applitools takes a visual approach to software testing, leading in automated visual testing and monitoring for web and mobile applications. It focuses on the visual accuracy of applications across different devices, browsers, and screen sizes. By using AI and machine learning, Applitools can detect visual bugs and discrepancies that traditional functional testing might miss. This represents a significant shift from traditional DOM-based testing and stands to benefit greatly from the integration of multimodal AI models.

4.8. QA Wolf
QA Wolf distinguishes itself by taking a novel approach to automated testing. It converts user traffic into automated functional tests, incorporating analytics features to prioritize test cases. This concept, while used in test data and virtualization, is relatively new to test automation. QA Wolf's AI-native approach is designed to deliver rapid, scalable, and maintainable tests. These tools demonstrate how scriptless automation has evolved, integrating AI to streamline the testing process, reduce maintenance, and improve the adaptability of tests to changes in applications.

5. Integration with CI/CD Pipelines
One of the key advantages of scriptless automation is its seamless integration with CI/CD pipelines. This integration is achieved through plugins or API interfaces that allow CI/CD tools to trigger test execution automatically. Key aspects include pipeline configuration, environment management, parallel execution, and continuous feedback. AI agents can take this one level up by:

5.1. Self-Configuring Pipelines
AI agents can automatically configure and adjust CI/CD pipelines to optimize test execution based on the latest code changes and test results. This reduces the manual effort required for pipeline management and ensures that testing remains aligned with development objectives.

5.2. Intelligent Resource Management
AI agents manage resources intelligently, optimizing the use of infrastructure for parallel test execution and ensuring that tests are run efficiently and effectively. These agents can predict resource requirements and allocate them dynamically.

5.3. Continuous Learning and Improvement
AI agents leverage continuous feedback from test results to refine their strategies and execution plans. This learning process ensures that testing becomes progressively more effective and less resource-intensive over time.

6. Challenges and Limitations
Despite its advantages, scriptless automation has certain limitations:

6.1. Limited Customization
Scriptless automation tools are generally optimized for common testing scenarios. However, complex or highly specialized test cases may require custom scripts or extensions.

6.2. Performance Overhead
The abstraction layers in scriptless tools can introduce performance overhead, particularly during test execution.

6.3. Tool Ecosystem Lock-In
Organizations may face challenges related to vendor lock-in, as migrating test cases from one tool to another can be difficult.

6.4. Learning Curve for Technical Users
Technical users accustomed to traditional scripting may find the transition to scriptless tools challenging.

7. Future Directions
Scriptless automation is poised to evolve with advancements in AI and ML. Future tools may incorporate AI-driven test case generation, self-healing tests, and enhanced analytics for predictive defect identification. Future AI agents will have even more advanced decision-making capabilities, enabling them to handle increasingly complex testing scenarios with minimal human intervention. AI agents will be applied to a broader range of use cases, including exploratory testing, security testing, and performance optimization, further expanding their utility in software development.

8. Conclusion
Scriptless automation represents a significant shift in the approach to test automation, offering technical and business benefits by reducing dependency on scripting, enhancing collaboration, and accelerating test case creation. AI agent-driven scriptless automation takes it one level up, offering unparalleled efficiency, fault tolerance, self-healing, stability, adaptability, and intelligence. While challenges remain, the potential benefits of deploying autonomous AI agents in testing processes are immense. However, it is not a one-size-fits-all solution. Organizations must carefully consider their specific needs before adopting scriptless automation.

References

[1] Aho Pekka et al., "Applying Scriptless Test Automation on Web Applications from The Financial Sector," *In Proceedings of the 25th Conference on Software Engineering and Databases (JISBD 2021)*, 2021. [Google Scholar] [Publisher Link]

[2] Ganesh Gatla, Kanchan Gatla, and Balaji Vishwanath Gatla, "Codeless Test Automation for Development QA," *American Academic Scientific Research Journal for Engineering, Technology, and Sciences*, vol. 91, no. 1, pp. 28-35, 2023. [Google Scholar] [Publisher Link]

[3] Tanja E.J. Vos et al., "TESTAR-Scriptless Testing Through Graphical User Interface," *Software Testing, Verification and Reliability*, vol. 31, no. 3, pp. 1-46, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[4] Takamasa Tanaka, Hidekazu Niibori, Li Shiyingxue, Shimpei Nomura, Tadayoshi Nakao, Kazuhiko Tsuda, "Selenium based Testing Systems for Analytical Data Generation of Website User Behavior," *In 2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pp. 216-221, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[5] Muneyoshi Iyama et al., "Automatically Generating Test Scripts for GUI Testing," *2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, Västerås, Sweden, pp. 146-150, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[6] José Campos et al., "Continuous Test Generation: Enhancing Continuous Integration with Automated Test Generation," *In Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*, New York, USA, pp. 55-66, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[7] Gordon Fraser, and Andrea Arcuri, "A Large-Scale Evaluation of Automated Unit Test Generation Using Evosuite," *In Proceedings PACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 24, no. 2, pp. 1-42, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[8] Lin Cheng et al., "GUICat: GUI Testing as A Service," *In Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, Singapore, pp. 858-863, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[9] Maurizio Leotta et al., "Robula$^+$: An Algorithm for Generating Robust Xpath Locators for Web Testing," *Journal of Software: Evolution and Process*, vol. 28, no. 3, pp. 177-204, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[10] Atif Memon, Ishan Banerjee, and Adithya Nagarajan, "GUI Ripping: Reverse Engineering of Graphical User Interfaces for Testing," *In Proceedings of the 10th Working Conference on Reverse Engineering*, Victoria, Canada, pp. 1-260, 2003. [Google Scholar] [Publisher Link]

[11] N. Nyman, "Using Monkey Test Tools - How to Find Bugs Cost-Effectively Through Random Testing," *Software Testing & Quality Engineering*, pp. 18-21, 2000. [Google Scholar]

[12] Mirella Martínez, "Towards Automated Testing of The Internet of Things: Results Obtained with The TESTAR Tool," *In Proceedings Leveraging Applications of Formal Methods, Verification and Validation. Distributed Systems*, Springer, Cham, pp. 375-385, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[13] Tanja E. J. Vos, "Evolutionary Testing for Complex Systems," ERCIM News, 2009. [Google Scholar] [Publisher Link]

[14] Ranorex, Functional UI Test Automation with Intelligent Test Design, 2024. [Online] Available: https://www.ranorex.com/

[15] Selenium, Selenium automates browsers. That's it!, 2024. [Online] Available: https://www.selenium.dev/

[16] RaiMan, Automate What You See on A Computer Monitor, Sikulix, 2024. [Online] Available: http://www.sikulix.com/

[17] Vivien Chinnapongse et al., "Model-Based Testing of GUI-Driven Applications," *Software Technologies for Embedded and Ubiquitous Systems, Lecture Notes in Computer Science*, pp. 203-214, vol. 5860, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[18] Børge Haugset, and Geir Kjetil Hanssen, "Automated Acceptance Testing: A Literature Review and an Industrial Case Study," *Agile 2008 Conference*, Toronto, ON, Canada, pp. 27-38, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[19] Faezeh Khorram, Jean-Marie Mottu, and Gerson Sunyé, "Challenges & Opportunities in Low-Code Testing," *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, no. 70, pp. 1-10, 2020. [CrossRef] [Google Scholar] [Publisher Link]