International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 06 Issue: 12 | Dec - 2022 SJIF Rating: 7.185 ISSN: 2582-3930

Mastering Object Healing Techniques for UI Automation: A Comprehensive Guide

Author 1

Dr. Ranjith Gopalan PhD,

Principal consultant, Cognizant

Email: ranjith.gopalan@gmail.com

Author 2

Mr. Arjun Bharadwaj Thyagarajan. MCA,

Principal Consultant, Cognizant

Email: tarjun.bharadwaj@gmail.com,

Abstract

Object healing plays a crucial role in the realm of UI tests, serving as a fundamental mechanism that enhances the resilience and stability of automated testing frameworks. As applications undergo constant changes, whether through updates in user interfaces or the addition of new features, the elements within those interfaces often change in ways that can disrupt automated tests. Object healing techniques address these disruptions by allowing test scripts to adapt to changes in UI elements, ensuring that tests remain valid and effective. This adaptability is particularly significant in dynamic environments, where the cost of maintaining outdated tests can be substantial.

This paper provides a comprehensive overview of the importance of object healing in UI tests. It explores the fundamentals of object recognition and examines various techniques for effective object recognition. Additionally, the paper delves into self-healing techniques using Healenium, a tool specifically designed to support UI applications built on React. Furthermore, it discusses the optimization of machine learning algorithms to enhance object healing in UI tests, and outlines best practices for object healing in continuous integration environments. The paper concludes with recommendations and insights into future trends in this rapidly evolving field.

Keywords: UI automation, Element recognization, self-healing techniques, Healenium, Object healing, decision tree algorithm

Introduction

Object healing is an essential concept in the realm of UI automation, particularly as applications evolve and user interfaces change over time. For researchers and students focusing on UI automation tests, grasping the nuances of object healing is critical for maintaining robust and reliable test suites. Object healing involves the process of automatically identifying and adapting to changes in UI elements, ensuring that test scripts continue to function even when the underlying application undergoes modifications. This adaptive capability is vital for sustaining the efficiency and effectiveness of automated testing, minimizing maintenance efforts, and enhancing the overall testing experience.

As applications become more complex, particularly in web and mobile environments, the need for advanced object healing techniques becomes apparent. Researchers should explore the various strategies that can be employed to effectively handle the dynamic nature of UI elements. For instance, in web applications, leveraging unique attributes or hierarchies can help automate the identification of elements even after significant changes have occurred. If html files are not developed with proper class elements and each deployment, recognization keeps changing or if any new elements introduced, that impact old element recognization, then automatic identification of objects using self-healing techniques with Healenium will help. Paper talks this very detailed with architecture flow

Integrating artificial intelligence into object healing processes represents a frontier that researchers should investigate. AI can enhance the decision-making capabilities of object healing algorithms, allowing them to learn from historical data and adapt to new patterns of UI changes. By employing machine learning techniques, automated tests can become more resilient and less reliant on rigid definitions of UI elements. This integration not only streamlines the healing process but also reduces the overall time spent on test maintenance, facilitating a more agile approach to software development and testing.

Methodology

Overview of Object Healing Techniques

Object healing techniques are essential for maintaining the reliability and effectiveness of UI automation tests, particularly in dynamic environments where elements frequently change. These techniques involve the identification and correction of broken object references, ensuring that tests can continue to function even when the underlying application undergoes modifications. Researchers and students focusing on UI automation will find that mastering these techniques not only enhances test stability but also contributes to more efficient testing processes overall.

In the realm of web applications, object healing strategies must adapt to the unique challenges presented by browser variability and element behavior. Advanced techniques such as XPath and CSS selector refinement, along with the use of visual recognition algorithms, can significantly improve the resilience of automated tests. By employing these strategies, researchers can develop more sophisticated automation frameworks that effectively handle changes in the user interface, thus minimizing maintenance efforts and maximizing test coverage.

The integration of artificial intelligence in object healing processes has opened new avenues for achieving robustness in test automation. By utilizing machine learning algorithms, researchers can analyze patterns in UI changes and predict potential breakages before they occur. This proactive approach not only reduces the time spent on manual adjustments but also contributes to a more streamlined testing lifecycle. The exploration of AI applications in object healing will be crucial for those aiming to stay at the forefront of test automation innovations.

Self-healing techniques with Healenium

Healenium automatically detects and fixes test failures caused by changes in the UI, like element IDs or class names. It uses advanced algorithms to analyze test results, identify the root cause of failures, and make necessary adjustments so the tests continue to work correctly.

Healenium includes the following services:

- postgres-db (PostgreSQL database to store reference selector / healing / report / DOM)
- hlm-proxy (Proxy client between Selenium server and application)
- hlm-backend (CRUD service)
- selector imitator (Convert healed locator to convenient format)
- selenium-grid (Selenium server)

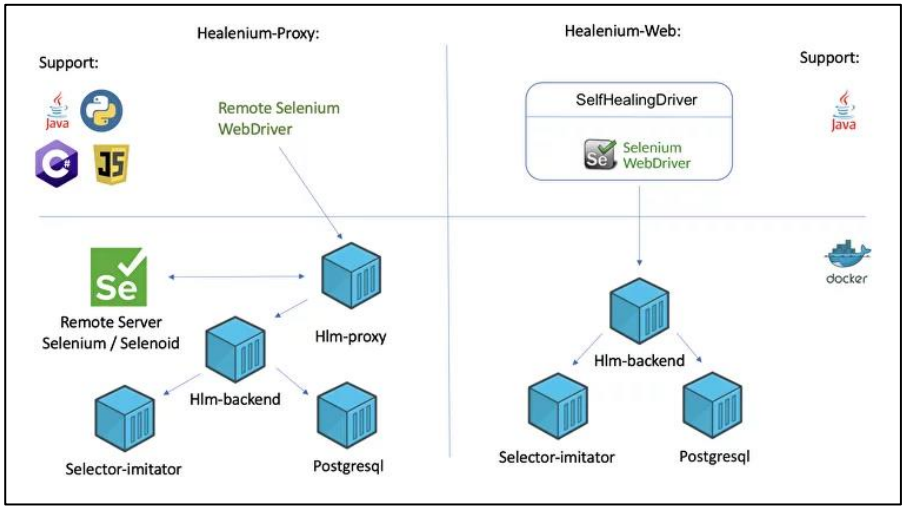

Figure 1: Healanium architecture and shows self-healing driver.

Role of AI and machine learning algorithms for object healing

The role of artificial intelligence (AI) in object healing has emerged as a transformative element in the realm of UI automation. Traditional object healing techniques often rely on predefined rules and manual interventions to address changes in user interface elements. However, AI introduces a dynamic approach that enhances the adaptability and efficiency of object healing processes. By employing machines learning algorithms, AI can analyze patterns in UI changes, enabling automated identification and mapping of elements even when their attributes are modified. This capability significantly reduces the time and effort required for test maintenance, leading to more robust automated testing frameworks.

One of the most significant advantages of integrating AI into object healing is its ability to learn from historical data. AI systems can utilize past interactions with UI elements to predict future changes, allowing for proactive adjustments in test scripts. This predictive capability not only improves accuracy but also minimizes the occurrence of false positives in automated tests. Additionally, AI can continuously evolve its understanding of the UI by processing real-time data, ensuring that the object healing strategies remain relevant as applications undergo

frequent iterations. Researchers can leverage this adaptability to create more resilient testing environments that can withstand rapid changes in application design.

Healenium uses a machine learning algorithm to enhance the stability of Selenium-based automated tests. Here is a detailed explanation of how it works:

How Healenium Works

1. **Page State Analysis**: Healenium analyzes the current state of the web page to detect any changes in the UI, such as updated element IDs or class names.
2. **Self-Healing Mechanism**: When a test failure occurs due to a change in the UI (e.g., NoSuchElement exception), Healenium triggers its self-healing mechanism. It uses a machine learning algorithm to find a new locator that matches the updated element1.
3. **Tree-Comparing Algorithm**: The self-healing process involves a tree-comparing algorithm that searches the current DOM state for the best subsequence and generates a new CSS locator.
4. **Integration with Selenium**: Healenium overrides the Selenium WebDriver's findElement method. If it catches a NoSuchElement exception, it triggers the tree-comparing algorithm to start the self-healing process1.
5. **Database Interaction**: The back-end part of Healenium uses a PostgreSQL database to store old and new locator values, along with related information like DOM page, method name, class name, and screenshots.
6. **Reporting**: After the test run, Healenium provides detailed reporting with the fixed locators and screenshots. It also includes plugins for Maven and Gradle to generate reports with healing results1.

Benefits of Healenium

- **Reduces Maintenance Effort**: By automatically fixing test failures caused by UI changes, Healenium minimizes the time and effort required to maintain automated tests.
- **Improves Test Stability**: Ensures that automated tests remain stable even when there are changes in the UI.
- **Detailed Reporting**: Provides comprehensive reports with fixed control values and screenshots, enhancing visibility and debugging.

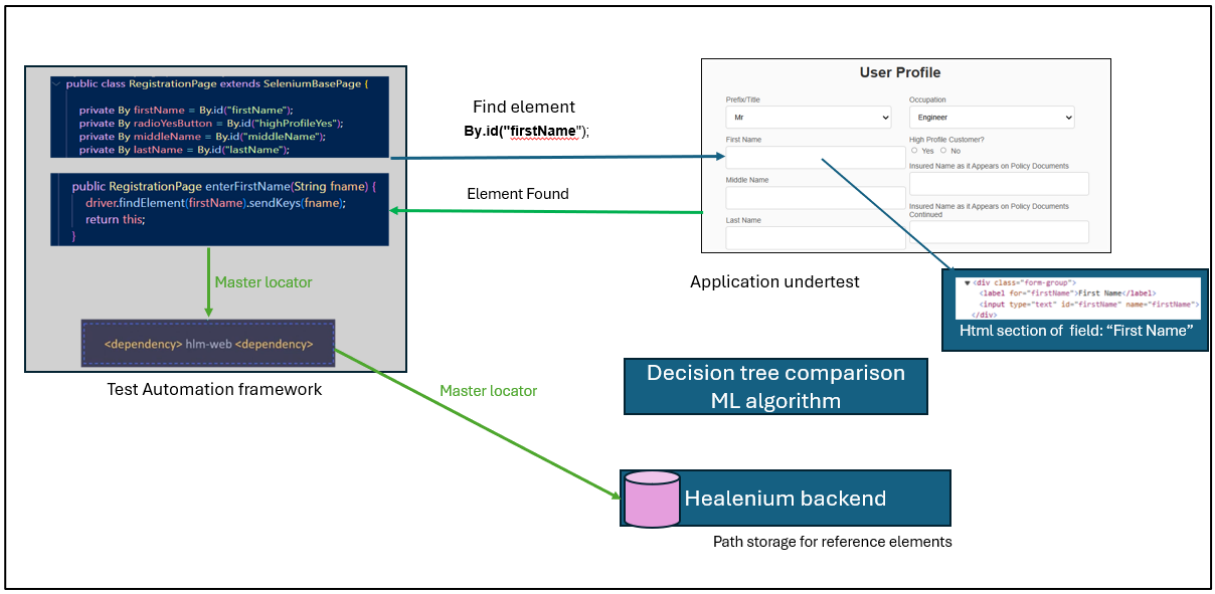

Figure 2: Healanium architecture that store master locator while baselining the locator information.

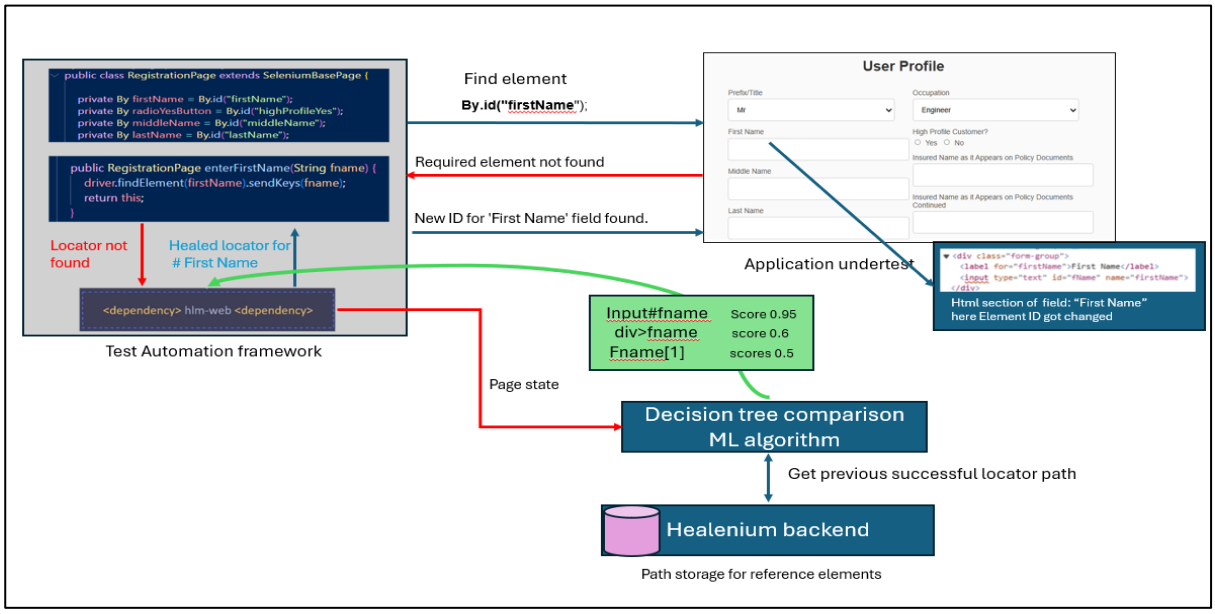

Figure 3: Healanium architecture that does auto healing is baseline locator has changes. This tree comparison decision is done by a decision tree algorithm.

Results

This paper presents the Healenium architecture, which enables reliable and self-healing Java Selenium automation frameworks. Automated scripts are developed for validating a web application where user profiles must be added to issue a home policy. This page is called the "Insured" page. To test the Healenium capability, an automated test is run, and the master locators are captured. Then, the application is redeployed by changing the locator's name. Here, particularly the "First Name" and "Last Name" properties got changed. However, using the Healenium capability, the locator was automatically healed, and the execution went successful, providing a report of the changes.

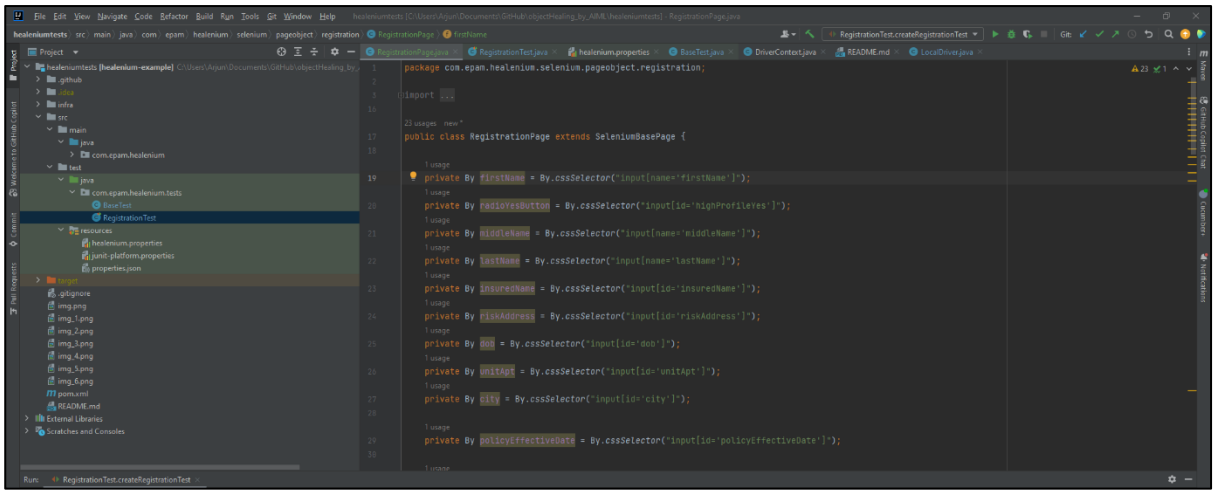

Figure 4: Healanium framework and automation script for the insured page testing

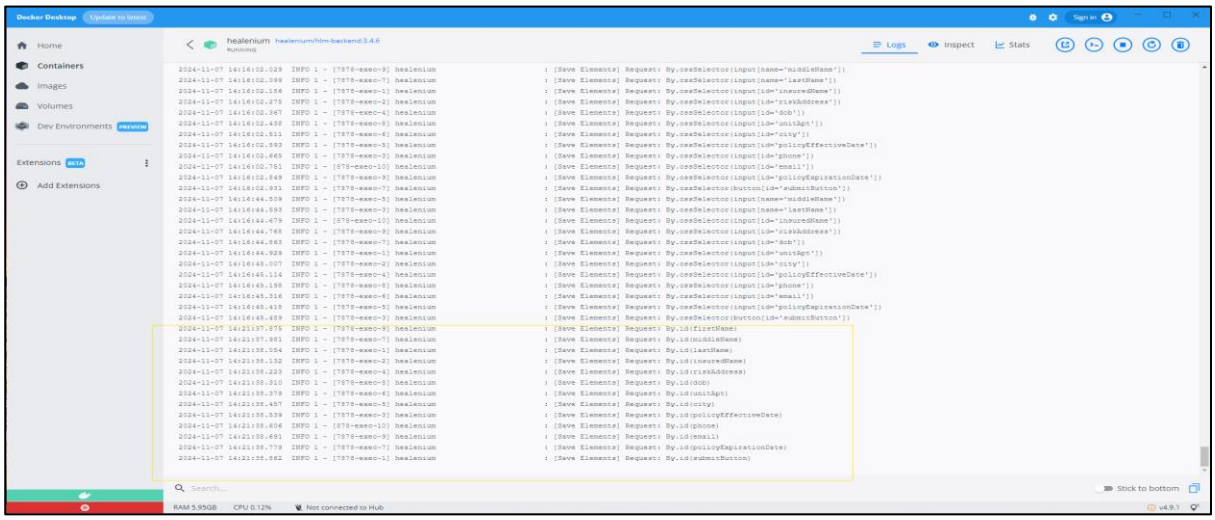

Figure 5: Healanium captures the locator and keep as master locator in the docker desktop.

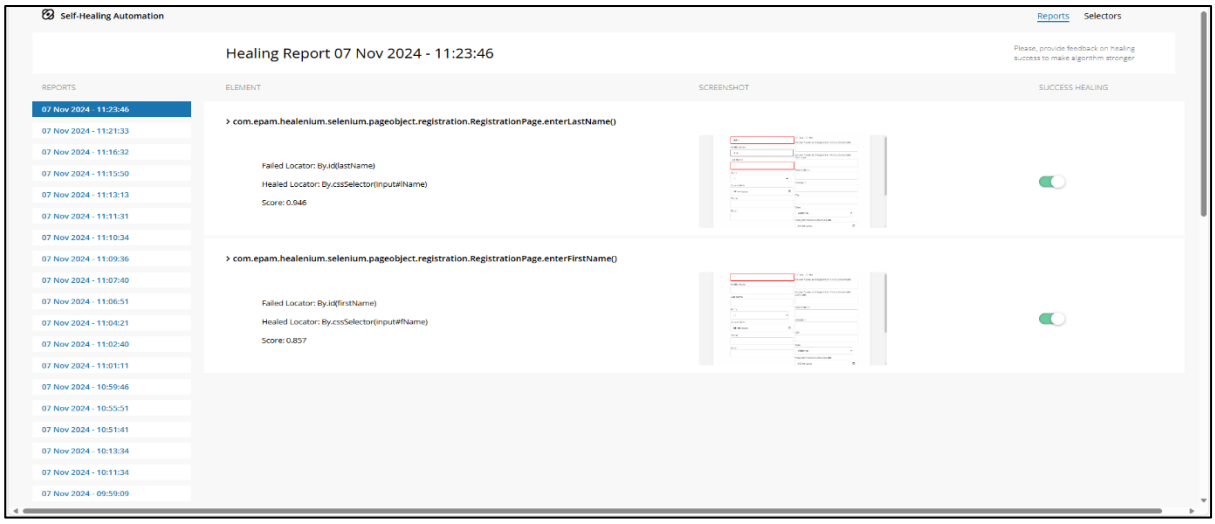

Figure 6: Locators for the field “First Name” and “Last Name” got changed and Healanium framework captured successfully and healed it.

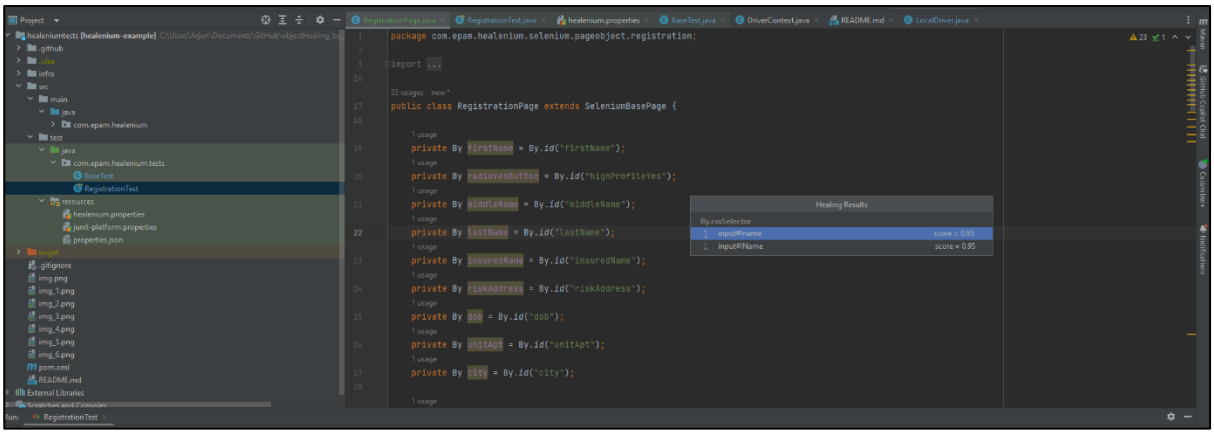

Figure 7: Please see the decision tree algorithm score for the field: Last name". It is 0.95

Discussion

Here are papers about optimization of object healing to make smooth running of automated UI scripts on desktop and mobile. Paper talks about enhancing AI techniques and implementing these techniques in CICD pipelines.

Optimize Machine Learning for Enhanced Object Healing in UI Tests

The integration of AI and machine learning algorithms in object healing techniques has demonstrated significant improvements in the robustness and efficiency of UI automation frameworks. The ability of AI-powered systems to adapt to UI changes, predict future locator failures, and dynamically update test scripts has reduced the time and effort required for manual test maintenance.

Training Models for Object Recognition

Constructing effective object recognition models is pivotal for bolstering the performance of object healing methods within UI automation. As automated testing systems increasingly depend on machine learning algorithms to identify and engage with UI elements, researchers and students must grasp the nuances of model training. This endeavor entails curating apt datasets that encapsulate a wide spectrum of user interfaces, guaranteeing the resultant models possess the resilience to recognize diverse elements across an array of applications, be it web or mobile.

Data preparation is a crucial step for training effective object recognition models. Researchers need to curate high-quality datasets that cover various scenarios and edge cases. This involves annotating images with labels corresponding to the UI elements. A well-prepared dataset boosts the model's accuracy and adaptability, which is vital in continuous integration setups where UI changes are frequent.

The choice of algorithms significantly impacts the performance of object recognition models. Researchers should explore various machine learning frameworks, such as Convolutional Neural Networks (CNNs), which have demonstrated exceptional performance in image recognition tasks. By experimenting with different architectures and hyperparameters, teams can optimize models for specific applications, tailoring their object healing strategies to accommodate unique UI frameworks and design patterns.

Evaluating Machine Learning Solutions in Object Healing

Rigorous evaluation of machine learning solutions is essential to ensure the efficacy of object healing techniques in UI automation.

Evaluating machine learning solutions for object healing demands a deep understanding of UI automation's unique needs and challenges. As researchers and students explore this field, they must consider the complexities of object

healing and how machine learning can boost effectiveness. Machine learning can improve the accuracy and efficiency of identifying and recovering objects, especially in dynamic environments with frequent UI changes. By analyzing diverse machine learning models, researchers can pinpoint the best approaches for different UI automation contexts.

A key aspect in evaluating machine learning solutions is the model's ability to generalize across various applications and frameworks. The model should not only perform well on a training dataset but also demonstrate adaptability to new or unseen UI elements. This adaptability is essential for maintaining the integrity of automated tests as applications evolve. Researchers should explore techniques such as transfer learning and domain adaptation to enhance the generalization capabilities of their models, ensuring effective object healing across diverse platforms, including web and mobile applications.

Assessing the model's real-time performance, particularly within continuous integration setups, is crucial. Evaluating the speed and efficiency of machine learning algorithms is pivotal, as delays in object healing can impede the testing process. Researchers should rigorously benchmark performance under diverse conditions, scrutinizing how different algorithms respond to evolving UI elements during active test runs. This comprehensive evaluation will pinpoint the most efficient solutions capable of seamless integration into CI/CD pipelines, accelerating feedback cycles and enhancing overall test automation effectiveness.

Best Practices for Object Healing in Continuous Integration Environments

Integrating robust object healing techniques within continuous integration environments is paramount for ensuring the reliability and scalability of UI automation frameworks.

Integrating Object Healing in CI/CD Pipelines

Integrating object healing into Continuous Integration/Continuous Deployment (CI/CD) pipelines represents a significant advancement in the realm of UI automation testing. As applications evolve and undergo frequent updates, the stability of automated tests can be jeopardized by changes in the user interface. Object healing techniques serve to dynamically adapt to these changes, ensuring that tests remain robust and reliable. In a CI/CD environment, where rapid iterations and deployments are the norm, embedding object healing into the pipeline can significantly reduce maintenance overhead and enhance test resilience.

The first step in integrating object healing into CI/CD pipelines is to establish a framework that supports automated test execution and healing capabilities. This involves selecting a suitable test automation tool that incorporates object healing features. Tools equipped with AI-driven algorithms can intelligently analyze failures and apply healing strategies based on the nature of the changes detected. Researchers and practitioners should focus on evaluating tools that not only provide healing functionalities but also seamlessly integrate with existing CI/CD systems, ensuring that the entire process—from code is committed to deployment—is streamlined.

Once the right tools are in place, it is crucial to define clear object healing strategies tailored to the specific needs of the application under test. Advanced techniques, such as leveraging machine learning algorithms, can significantly enhance the effectiveness of object healing. By training models on historical test data, teams can develop predictive capabilities that anticipate UI changes and adapt tests accordingly. This proactive approach minimizes the risk of test failures due to UI changes and fosters a more reliable testing environment, ultimately leading to higher quality software releases.

Monitoring and Reporting Object Healing Results

Keeping an eye on and reporting the results of object healing is key to making sure UI automation tests stay useful and trustworthy. As researchers and students dig into the details of object healing, knowing how to track and study the outcomes of these processes is a must. Good monitoring lets teams spot problems fast, check how well the healing tools are working, and decide what changes to make to their automated testing setups.

Monitoring object healing results usually involve logging. By recording details of healing attempts, including the original and fixed properties of UI elements, researchers can spot trends over time. This data gives insight into UI changes and helps assess how well different healing methods work. The logs can also include visual reports showing the UI before and after healing, making it easier for teams to understand the impact.

Tracking metrics is key to understanding how well object healing is working. Things like success rate, healing time, and false results can show how effective it is. Researchers should set a standard to compare over time. Regularly checking these numbers helps find areas to improve, making UI tests more reliable.

Future implementations

Despite the promising developments in automated UI testing and object healing, several key challenges remain that require further research and innovation:

Generalization and Adaptability: Existing object healing techniques are often tailored to specific applications or UI frameworks, limiting their ability to generalize across diverse environments. Researchers should focus on developing models that can adapt to new or unseen UI elements, enhancing the long-term viability and scalability of automated testing solutions.

Real-time Performance: The speed and efficiency of object healing algorithms are critical, particularly within continuous integration setups where test execution must be timely.

Predictive Object Healing

Predictive Object Healing is an emerging trend in the realm of UI automation that leverages advanced machine learning techniques to anticipate and rectify issues with UI elements before they lead to test failures. In traditional UI automation, tests often break due to changes in the application's user interface, such as modifications to element properties or structure. Predictive Object Healing aims to mitigate this problem by utilizing historical data and contextual information to forecast potential changes and adapt the automated tests accordingly. This proactive approach not only enhances the reliability of automated tests but also significantly reduces maintenance efforts.

The core principle behind Predictive Object Healing is the analysis of previous test executions and the contextual understanding of UI components. By examining patterns in how UI elements change over time, machine learning algorithms can predict which elements are likely to become unstable or change. For instance, if a button's identifier has historically changed after a particular type of update, the predictive model can flag this button for monitoring. This predictive analysis allows automation frameworks to prepare for changes, thus ensuring that test scripts remain valid and functional despite modifications in the UI.

Implementing Predictive Object Healing requires an ecosystem that integrates data collection, analysis, and real-time feedback into the UI automation process. Automation tools must gather data from various test runs and user interactions to build a comprehensive model of how UI elements behave. This data can include metrics such as element visibility, interaction frequency, and historical changes. By feeding this information into machine learning

algorithms, automation frameworks can generate insights that guide test updates and adjustments, effectively healing the automation scripts as changes occur in the application.

Integrating AI for Smart Healing

Integrating artificial intelligence into the domain of smart healing represents a significant advancement in the field of UI automation. As applications become increasingly complex, the need for efficient error detection and resolution mechanisms has grown. Traditional debugging and healing processes can be time-consuming and often require human intervention. By leveraging AI, developers can create systems that not only identify problems but also autonomously implement effective solutions, thereby streamlining the user experience and enhancing overall application reliability.

The integration of natural language processing (NLP) further enhances AI's ability to facilitate smart healing. By enabling AI systems to interpret user feedback and system logs in human language, developers can gain insights into user experiences that were previously difficult to quantify. This understanding not only aids in the immediate healing process but also informs future design decisions, creating a feedback loop that contributes to the overall improvement of user interfaces. Consequently, as AI systems become more adept at understanding context and intent, their ability to preemptively address issues will significantly increase.

Conclusion

This paper provided a thorough examination of the pivotal role of object healing in UI test automation. It delved into the core principles of object recognition, exploring diverse techniques for robust object identification. Moreover, the paper delved into the self-healing capabilities of Healenium, a tool tailored to support UI applications built on React. Additionally, it explored the optimization of machine learning algorithms to amplify object healing in UI testing and outlined best practices for embedding object healing within continuous integration pipelines. The paper culminates with forward-looking recommendations and insights into the rapidly evolving landscape of this domain.

Overall, this comprehensive guide equips researchers and practitioners with a deep understanding of object healing and its transformative impact on UI test automation.

Reference

Ahmed, A. (2014, January 1). Test automation for Graphical User Interfaces: A review. https://doi.org/10.1109/wccais.2014.6916544

Aho, P., & Vos, T E J. (2018, April 1). Challenges in Automated Testing Through Graphical User Interface., 1, 118-121. https://doi.org/10.1109/icstw.2018.00038

Alégroth, E., & Feldt, R. (2017, January 24). On the long-term use of visual gui testing in industrial practice: a case study. Springer Science+Business Media, 22(6), 2937-2971. https://doi.org/10.1007/s10664-016-9497-6

Amershi, S., Weld, D., Vorvoreanu, M., Fourney, A., Nushi, B., Collisson, P., Suh, J., Iqbal, S T., Bennett, P N., Inkpen, K., Teevan, J., Kikin-Gil, R., & Horvitz, E. (2019, April 29). Guidelines for Human-AI Interaction. https://doi.org/10.1145/3290605.3300233

Åström, J., Reim, W., & Parida, V. (2022, January 20). Value creation and value capture for AI business model innovation: a three-phase process framework. Springer Science+Business Media, 16(7), 2111-2133. https://doi.org/10.1007/s11846-022-00521-z

Avcioglu, A., & Demirer, M. (2015, November 1). Implementation of system testing automatization on computer aided systems for hardware and software. , 127-133. https://doi.org/10.1109/autest.2015.7356478

Bureš, M. (2015, June 25). Metrics for automated testability of web applications. https://doi.org/10.1145/2812428.2812458

Cheng, Y., & Elsayed, E A. (2020, August 20). Design of optimal sequential hybrid testing plans. Taylor & Francis, 53(7), 830-841. https://doi.org/10.1080/24725854.2020.1805828

Coppola, R., Ardito, L., & Torchiano, M. (2019, August 8). Fragility of layout-based and visual GUI test scripts: an assessment study on a hybrid mobile application. https://doi.org/10.1145/3340433.3342824

Coppola, R., Ardito, L., Morisio, M., & Torchiano, M. (2020, September 1). Mobile Testing: New Challenges and Perceived Difficulties From Developers of the Italian Industry. IEEE Computer Society, 22(5), 32-39. https://doi.org/10.1109/mitp.2019.2942810

Daniel, B., Luo, Q., Mirzaaghaei, M., Dig, D., Marinov, D., & Pezzé, M. (2011, July 17). Automated GUI refactoring and test script repair. https://doi.org/10.1145/2002931.2002937

Gebreyohannes, S., Karimoddini, A., & Homaifar, A. (2020, August 24). Applying Model-Based Systems Engineering to the Development of a Test and Evaluation Tool for Unmanned Autonomous Systems. https://doi.org/10.1109/syscon47679.2020.9275894

Ghahremani, S., & Giese, H. (2020, February 27). Evaluation of Self-Healing Systems: An Analysis of the State-of-the-Art and Required Improvements. Multidisciplinary Digital Publishing Institute, 9(1), 16-16. https://doi.org/10.3390/computers9010016

Gîrju, R. (2021, May 14). Adaptive Multimodal and Multisensory Empathic Technologies for Enhanced Human Communication. Cornell University. https://doi.org/10.48550/arXiv.2110.

Grechanik, M., Xie, Q., & Fu, C. (2009, January 1). Creating GUI Testing Tools Using Accessibility Technologies. https://doi.org/10.1109/icstw.2009.31

Guoqingx@ics.uci.edu, G X U O C I. (2013, October 28). Resurrector. https://dl.acm.org/doi/10.1145/2544173.2509512

Huang, F., Li, G., Zhou, X., Canny, J., & Yang, L. (2021, January 1). Creating User Interface Mock-ups from High-Level Text Descriptions with Deep-Learning Models. Cornell University. https://doi.org/10.48550/arxiv.2110.07775

Imtiaz, J., Iqbal, M Z., & Khan, M U. (2020, September 28). An automated model-based approach to repair test suites of evolving web applications. Elsevier BV, 171, 110841-110841. https://doi.org/10.1016/j.jss.2020.110841

Li, A., Qin, Z., Chen, M., & Liu, J. (2014, June 1). ADAutomation: An Activity Diagram Based Automated GUI Testing Framework for Smartphone Applications. , 68-77. https://doi.org/10.1109/sere.2014.20

Li, S. (2020, October 1). The Trend and Characteristic of AI in Art Design. IOP Publishing, 1624(5), 052028-052028. https://doi.org/10.1088/1742-6596/1624/5/052028

Long, Z., Wu, G., Chen, X., Chen, W., & Wei, J. (2020, November 8). WebRR: self-replay enhanced robust record/replay for web application testing. https://doi.org/10.1145/3368089.3417069

Lowry, O., Rosebrough, N., Farr, A., & Randall, R. (1951, November 1). PROTEIN MEASUREMENT WITH THE FOLIN PHENOL REAGENT. Elsevier BV, 193(1), 265-275. https://doi.org/10.1016/s0021-9258(19)52451-6

Martiniello, N., Asuncion, J V., Fichten, C S., Jorgensen, M., Havel, A., Harvison, M., Legault, A., Lussier, A., & Vo, C. (2020, December 1). Artificial intelligence for students in postsecondary education. Association for Computing Machinery, 6(3), 17-29. https://doi.org/10.1145/3446243.3446250

Moreira, R M., Paiva, A C R., Nabuco, M., & Memon, A M. (2017, March 2). Pattern-based GUI testing: Bridging the gap between design and quality assurance. Wiley, 27(3). https://doi.org/10.1002/stvr.1629

Morgado, I C., & Paiva, A C R. (2015, November 1). Testing Approach for Mobile Applications through Reverse Engineering of UI Patterns. https://doi.org/10.1109/asew.2015.11

Nagabushanam, D S., S, S D., Dharinya, V S., Roopa, N S., & Arun, A. (2022, January 1). A Review on the Process of Automated Software Testing. Cornell University. https://doi.org/10.48550/arXiv.2209.

Naqvi, M A., Astekin, M., Malik, S., & Moonen, L. (2021, March 1). Adaptive Immunity for Software: Towards Autonomous Self-healing Systems. https://doi.org/10.1109/saner50967.2021.00058

Patterson, D A., Brown, A., Broadwell, P., Candea, G., Chen, M Y., Cutler, J., Enriquez, P., Fox, A., Merzbacher, M., Oppenheimer, D., Sastry, N., Tetzlaff, W H., Traupman, J., Treuhaft, N., & Patterson, D A. (2002, January 1). Recovery Oriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies. http://dslab.epfl.ch/pubs/roc_vision.pdf

Petrovskaya, A., Pavlenko, D., Feofanov, K., & Klimov, V. (2020, January 1). Computerization of learning management process as a means of improving the quality of the educational process and student motivation. Elsevier BV, 169, 656-661. https://doi.org/10.1016/j.procs.2020.02.194

Radziwill, N., & Freeman, G. (2020, January 1). Reframing the Test Pyramid for Digitally Transformed Organizations. Cornell University. https://doi.org/10.48550/arXiv.2011.

Rezwana, J., & Maher, M L. (2022, February 24). Designing Creative AI Partners with COFI: A Framework for Modeling Interaction in Human-AI Co-Creative Systems. Association for Computing Machinery, 30(5), 1-28. https://doi.org/10.1145/3519026

Rikakis, T., Kelliher, A., Huang, J., & Sundaram, H. (2018, June 27). Progressive cyber-human intelligence for social good. Association for Computing Machinery, 25(4), 52-56. https://doi.org/10.1145/3231559

Romano, A., Song, Z., Grandhi, S., Yang, W., & Wang, W. (2021, May 1). An Empirical Analysis of UI-Based Flaky Tests. https://doi.org/10.1109/icse43902.2021.00141

Schilling, A., Madeira, K., Donegan, P., Sousa, K., Furtado, E., & Furtado, V. (2005, May 15). An integrated method for designing user interfaces based on tests. Association for Computing Machinery, 30(4), 1-5. https://doi.org/10.1145/1082983.1083280

Selay, E., Zhou, Z Q., Chen, T Y., & Kuo, F. (2018, September 25). Adaptive Random Testing in Detecting Layout Faults of Web Applications. World Scientific, 28(10), 1399-1428. https://doi.org/10.1142/s0218194018500407

Seng, L K., Ithnin, N., & Said, S Z M. (2018, September 28). The approaches to quantify web application security scanners quality: a review. , 8(38), 285-312. https://doi.org/10.19101/ijacr.2018.838012

Shin, M. (2005, January 14). Self-healing components in robust software architecture for concurrent and distributed systems. Elsevier BV, 57(1), 27-44. https://doi.org/10.1016/j.scico.2004.10.003

Sneha, K., & Malle, G M. (2017, August 1). Research on software testing techniques and software automation testing tools. https://doi.org/10.1109/icecds.2017.8389562

Stocco, A. (2019, April 1). How artificial intelligence can improve web development and testing. , 1-4. https://doi.org/10.1145/3328433.3328447

Stocco, A., Yandrapally, R., & Mesbah, A. (2018, October 26). Visual web test repair. https://doi.org/10.1145/3236024.3236063

Tirodkar, A A., & Khandpur, S S. (2019, May 1). EarlGrey: iOS UI Automation Testing Framework. https://doi.org/10.1109/mobilesoft.2019.00010

Yang, Q., Steinfeld, A., Rosé, C P., & Zimmerman, J. (2020, April 21). Re-examining Whether, Why, and How Human-AI Interaction Is Uniquely Difficult to Design. https://doi.org/10.1145/3313831.3376301

Yasmin, N., & Sitaraman, M. (2009, March 19). Compositional performance prediction exemplified using generic object finalization analysis. , 1-6. https://doi.org/10.1145/1566445.1566464

Yildirim, A S., Berker, E., & Kayakesen, M E. (2018, September 1). System Level Test Automation in UAV Development. https://doi.org/10.1109/autest.2018.8532551

Yu, Z., Fahid, F M., Menzies, T., Rothermel, G., Patrick, K., & Cherian, S. (2019, August 9). TERMINATOR: better automated UI test case prioritization. , 883-894. https://doi.org/10.1145/3338906.3340448

Zhao, J., Jin, Y., Trivedi, K S., Matias, R., & Wang, Y. (2014, January 1). Software rejuvenation scheduling using accelerated life testing. Association for Computing Machinery, 10(1), 1-23. https://doi.org/10.1145/2539118

Zhao, J., Wang, Y., Ning, G., Trivedi, K S., Matias, R., & Cai, K. (2013, July 16). A comprehensive approach to optimal software rejuvenation. Elsevier BV, 70(11), 917-933. https://doi.org/10.1016/j.peva.2013.05.010

Zimmerman, J., Oh, C., Yildirim, N., Kass, A., Tung, T., & Forlizzi, J. (2020, December 23). UX designers pushing AI in the enterprise. Association for Computing Machinery, 28(1), 72-77. https://doi.org/10.1145/3436954