International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Methods, Tools and Strategies of Cross Browser Testing a Web Application

Venkata Padma Kumar Vemuri

Santa Clara, USA
padma.vemuri@gmail.com

Abstract
Ensuring consistent user experiences across diverse web browsers is a persistent challenge. This paper reviews the current landscape of cross-browser testing, examining established methods, tools, and strategies—from manual techniques and automated frameworks to cloud-based platforms and CI/CD integrations. We highlight their strengths, limitations, and practical considerations. The review also anticipates the future role of AI and machine learning in predicting and preventing compatibility issues, optimizing test selection, and streamlining maintenance. By synthesizing current practices and emerging insights, this work informs both practitioners and researchers on improving the reliability, scalability, and efficiency of cross-browser testing.

Keywords: Cross-browser testing, testing tools, systematic literature review, web application.

INTRODUCTION

In today's dynamic and diverse digital environment, web applications must perform flawlessly across various browsers, operating systems, and devices. Users demand a consistent experience regardless of their chosen platform, and failing to meet these expectations can quickly lead to diminished trust, lower engagement, and revenue loss. Variations in browser rendering engines, JavaScript execution, and standards compliance create unique challenges that complicate achieving uniform functionality. Consequently, cross-browser testing becomes an essential phase in development, ensuring that design elements, interactive features, and performance metrics are reliably maintained across different environments. By rigorously testing applications across multiple browsers, developers can create more stable, inclusive, and user-friendly solutions, fostering improved brand reputation, accessibility, and sustainable growth.

METHODOLOGIES

A. Specification Transformation

Koopman et al. propose a method to transform program specifications to include browser navigation, which simplifies the testing process by allowing test engineers to focus on exceptions rather than general rules. This approach is implemented using the model-based test tool G∀ST, which helps identify errors in web applications by simulating browser navigation actions like back and forward buttons [12].

B. On-the-Fly Navigation Models

Song et al. introduce an approach that models web browser interactions using extended finite state machines (FSM). This method generates tests that account for browser-specific behaviors, ensuring that

navigation and functionality are consistent across different browsers [5].

C. Attribute and Tolerance-Based Testing

This system allows users to select specific web elements and attributes for comparison across browsers, using a defined tolerance level. It generates reports based on these comparisons, independent of language, which can be useful for international applications.

D. Model Comparison

Shi and Zeng utilize CRAWLJAX to create behavioral models of web applications in different browsers. By comparing these models, they can identify and locate incompatibility issues, reducing the workload for testing and repairs [6].

E. Image Processing Techniques

- Image processing methods involve capturing screenshots of web pages rendered in different browsers and comparing them to a baseline image. This approach helps identify visual differences that may not be apparent through DOM analysis alone [3][16].
- Techniques such as image segmentation and feature extraction are used to analyze specific regions of a web page. These regions are compared across browsers to detect inconsistencies [3][16].

F. Automation and Efficiency

Automated visual compatibility testing tools, such as the one described in X-BROT, reduce the time and cost associated with manual testing. These tools automate the process of comparing DOM elements and their visual representations across browsers [7].

Challenges in Cross-Browser Testing

A. Dynamic Component Identification

One of the significant challenges is identifying dynamic components in user interfaces, which can behave differently across browsers. This issue complicates the testing process and often leads to missed incompatibilities [1].

B. False positves and Negatives

Automated testing tools can generate false positives, where differences are flagged as issues even if they do not affect user experience. Conversely, false negatives occur when actual issues are not detected. This is particularly problematic in DOM-based analysis [3].

C. Non-Deterministic Events

Events such as asynchronous requests and timers can lead to inconsistencies that are difficult to detect with traditional testing methods. X-Check addresses this by focusing on DOM-mutated and layout-changed nodes, improving detection efficiency [9].

D. Browser-Specific Features

Various browsers may support unique features or interpret web standards in distinct ways, resulting in inconsistencies. To address these issues, developers often need to apply browser-specific adjustments, which can be both time-intensive and prone to errors [19].

While cross-browser testing methodologies have advanced significantly, challenges remain, particularly in automating the detection of dynamic and non-deterministic issues. The development of more sophisticated tools and techniques, such as those leveraging machine learning, holds promise for improving the accuracy and efficiency of cross-browser testing. However, the subjective nature of what constitutes an incompatibility, and the rapid evolution of web technologies continue to pose hurdles for developers and testers alike.

1) Strategies for Cross-Browser Testing

Cross-browser testing is a critical aspect of ensuring a consistent user experience across a diverse range of browsers, operating systems, and devices. This process combines manual and automated methodologies, leveraging various tools and best practices. The following strategies are widely adopted in the industry to achieve effective cross-browser testing.

a) Manual Testing on Multiple Browsers: Conduct tests on local machines or virtual machines (VMs) using the latest versions of popular browsers like Chrome, Firefox, Safari, Edge, and older versions of Internet Explorer to validate UI and functionality.

b) Using Virtual Machines and Containers**:** Tools like VMware, VirtualBox, and native OS virtualization (e.g., macOS Virtualization) allow teams to run various browser/OS combinations on a single physical machine. Platforms like Docker offer preconfigured containers with specific browser versions, providing a consistent testing environment that can be quickly deployed or terminated.

c) Cloud-Based Cross-Browser Testing Services: Services such as BrowserStack, Sauce Labs, CrossBrowserTesting, and LambdaTest provide instant access to a wide range of browser/OS/device combinations.

d) Automated Testing Frameworks and Tools: A versatile, language-agnostic framework for automating browser interactions across different browsers.

e) Regression Testing and Visual Comparisons:

- Visual Regression Tools: Tools like Percy and Applitools perform visual snapshot comparisons across browser versions to detect subtle style or layout inconsistencies.
- Baseline Comparisons: Automated tests verify the stability of previously supported browsers after each code change.

TABLE 1. POPULAR METHODS OF TESTING

Method	Recomended for
Manual Testing on Multiple Browsers	Small or MVP projects that need quick checks- Exploratory testing, UI/UX reviews- Ad-hoc or urgent compatibility checks
Virtual Machines & Containers	In-house teams needing specific legacy browser/OS combos- Highly controlled and reproducible environments
Cloud-Based Testing Services (BrowserStack, Sauce Labs)	Large-scale, complex applications requiring many browser/OS/device combos- Distributed/remote QA teams- CI/CD integration for continuous testing
Automated Frameworks (Selenium, Playwright, Cypress)	Applications under continuous development with frequent releases- Enterprise environments needing scalable, automated coverage- Complex user workflows and e-commerce flows
Visual/Regression Tools (Applitools, Percy)	Design/branding-sensitive projects- Ensuring pixel-perfect consistency across browsers- Front-end redesigns or framework migrations

MACHINE LEARNING IMPROVEMENTS

Machine learning (ML) can augment and enhance cross-browser testing processes in several keyways, making tests smarter, faster, and more reliable. While ML doesn't fully replace traditional testing tools and methods, it can streamline the testing workflow, provide deeper insights, and reduce the amount of human intervention needed. Here are some specific applications.

A. Machine Learning and Screenshot Similarity

One approach involves using machine learning algorithms combined with screenshot similarity analysis to detect XBIs. This method captures screenshots of web applications across different browsers and uses supervised learning to identify discrepancies that indicate incompatibilities [20].

B.Detection of UI Display Issues

- The OwlEye approach uses deep learning to model visual information from GUI screenshots, detecting display issues such as text overlap and missing images. It achieves 85% precision and 84% recall in detecting UI display issues, and 90% accuracy in localizing these issues [21][23].
- A machine learning model has been employed to visually compare user interface information with design specifications, identifying defects that do not visually match the design [12]. This approach helps in maintaining design consistency across different platforms.

C. Semantic Understanding and Automation

ActionBert, a pre-trained UI representation model, leverages visual, linguistic, and domain-specific features to understand UI components' functionality. It outperforms multi-modal baselines by up to 15.5% in various tasks, such as icon classification and UI component retrieval. The integration of ActionBert into the workflow not only enhances accuracy in identifying UI components but also streamlines the overall design review process, allowing for quicker iterations and improved user experiences [24].

D. Enhancing Usability and Interaction Analysis

- Machine learning models can analyze user interactions with UI elements, using interaction data to establish baseline values for UI elements. Abnormal interaction patterns can trigger reports or guidance, improving usability and user experience [12].
- A deep neural network-based method using symbol markers has been proposed to improve UI detection accuracy, facilitating better communication between designers and developers [25].

CONCLUSION

Cross-browser testing remains a cornerstone of modern web application development, ensuring users enjoy a seamless experience across diverse browsers, operating systems, and devices. The methodologies discussed—ranging from model-based testing and specification transformations to visual comparisons and automated tools—highlight the varied approaches available today. Each method brings unique benefits: systematic analysis through model-based techniques, precise rendering checks using image processing tools, and accelerated feedback loops enabled by automation and CI/CD integrations. Additionally, cloud-based platforms and containerized setups simplify the complexity of testing across different environments.

In conclusion, as the web continues to evolve, so must the tools and strategies for cross-browser testing. By adhering to proven practices, embracing modern automation, and harnessing the potential of ML-driven techniques, development teams can overcome inconsistencies, reduce maintenance burdens, and deliver a high-quality user experience across an ever-expanding array of platforms.

REFERENCES

1. Sabaren, L. N., Mascheroni, M. A., Greiner, C. L., & Irrazábal, E. (2018). A Systematic Literature Review in Cross-browser Testing. Journal of Computer Science and Technology. https://doi.org/10.24215/16666038.18.E03
2. Wu, G., He, M., Chen, W., Wei, J., & Zhong, H. (2021). X-Check: Improving Effectiveness and Efficiency of Cross-Browser Issues Detection for JavaScript-Based Web Applications. IEEE Transactions on Services Computing. https://doi.org/10.1109/TSC.2018.2860983
3. Saar, T., Dumas, M., Kaljuve, M., & Semenenko, N. (2016). Browserbite: cross-browser testing via image processing. Software - Practice and Experience. https://doi.org/10.1002/SPE.2387
4. Koopman, P., Achten, P., & Plasmeijer, R. (2008). Model-based testing of thin-client web applications and navigation input. https://doi.org/10.1007/978-3-540-77442-6_20
5. Song, B., Miao, H., & Chen, S. (2008). Modeling Web Browser Interactions and Generating Tests. Computational Intelligence and Security. https://doi.org/10.1109/CIS.2008.188
6. Shi, H., & Zeng, H. (2015). Cross-Browser Compatibility Testing Based on Model Comparison. https://doi.org/10.1109/CCATS.2015.34
7. Tanaka, H. (2019). X-BROT: Prototyping of Compatibility Testing Tool for Web Application Based on Document Analysis Technology. International Conference on Document Analysis and Recognition. https://doi.org/10.1109/ICDARW.2019.60126
8. Malla, P. K. (2014). Cross-browser web application testing tool.
9. Wu, G., He, M., Chen, W., Wei, J., & Zhong, H. (2021). X-Check: Improving Effectiveness and Efficiency of Cross-Browser Issues Detection for JavaScript-Based Web Applications. IEEE Transactions on Services Computing. https://doi.org/10.1109/TSC.2018.2860983
10. Koopman, P., Achten, P., & Plasmeijer, R. (2008). Model-based testing of thin-client web applications and navigation input. https://doi.org/10.1007/978-3-540-77442-6_20
11. Shi, H., & Zeng, H. (2015). Cross-Browser Compatibility Testing Based on Model Comparison. https://doi.org/10.1109/CCATS.2015.34
12. Sumaiya, P. K., Rajesh, V., Natarajan, D., Rohith, M. K., Jumi, B., Sangeetha, B., Basavana, G. S., & Sai, B. (2020). Machine learning analysis of user interface design.
13. Malla, P. K. (2014). Cross-browser web application testing tool.
14. Sabaren, L. N., Mascheroni, M. A., Greiner, C. L., & Irrazábal, E. (2018). A Systematic Literature Review in Cross-browser Testing. Journal of Computer Science and Technology. https://doi.org/10.24215/16666038.18.E03
15. Saar, T., Dumas, M., Kaljuve, M., & Semenenko, N. (2015). Browserbite: Cross-Browser Testing via Image Processing. arXiv: Software Engineering.
16. Semenenko, N., Dumas, M., & Saar, T. (2013). Browserbite: Accurate Cross-Browser Testing via Machine Learning over Image Features. International Conference on Software Maintenance. https://doi.org/10.1109/ICSM2013.88
17. Saar, T., Dumas, M., Kaljuve, M., & Semenenko, N. (2014). Cross-Browser Testing in Browserbite. https://doi.org/10.1007/978-3-319-08245-5_37
18. Choudhary, S. R., Versee, H., & Orso, A. (2010). A cross-browser web application testing tool. International Conference on Software Maintenance. https://doi.org/10.1109/ICSM.2010.5609728

19. Wu, G., He, M., Chen, W., Wei, J., & Zhong, H. (2021). X-Check: Improving Effectiveness and Efficiency of Cross-Browser Issues Detection for JavaScript-Based Web Applications. IEEE Transactions on Services Computing. https://doi.org/10.1109/TSC.2018.2860983
20. Barskar, N., & Patidar, C. P. (2016). A Survey on Cross Browser Inconsistencies in Web Application. International Journal of Computer Applications. https://doi.org/10.5120/IJCA2016908711
21. Paes, F. C. (2017). Automatic detection of cross-browser incompatibilities using machine learning and screenshot similarity: student research abstract. Symposium on Applied Computing. https://doi.org/10.1145/3019612.3019924
22. Liu, Z. (2020). Discovering UI display issues with visual understanding. Automated Software Engineering. https://doi.org/10.1145/3324884.3418917
23. Pathapati, P., Mohan, D., & Morrison, M. (2018). Utilizing a machine learning model to automatically visually validate a user interface for multiple platforms.
24. He, Z., Sunkara, S., Zang, X., Xu, Y., Liu, L., Wichers, N., Schubiner, G., Lee, R. B., Chen, J., & Arcas, B. A. y. (2020). ActionBert: Leveraging User Actions for Semantic Understanding of User Interfaces. arXiv: Computation and Language.
25. Park, J., Yun, Y.-S., Eun, S., Cha, S., So, S.-S., & Jung, J. (2018). Deep neural networks based user interface detection for mobile applications using symbol marker. Research in Adaptive and Convergent Systems. https://doi.org/10.1145/3264746.3264808