International Journal of Interdisciplinary Innovative Research & Development (IJIIRD)
ISSN: 2456-236X
Vol. 11 Special Issue 02 | 2026

Automated Detection of Reflow Accessibility Failures in Responsive Web Interfaces

Prof. Saurav V. Kolhe[1]
Department of Computer Science & Engineering, Padm. Dr. V. B. Kolte College of Engg., M.H, India

DOI: 10.5281/zenodo.19286155

ABSTRACT

Responsive web design is widely adopted to support multiple device types and viewport sizes. Despite its benefits, improper reflow implementation often results in accessibility failures, particularly for users who depend on keyboard-based assistive technologies. These failures may cause essential functionalities available in full-sized layouts to become inaccessible when the interface is reflowed for smaller viewports. Such issues, referred to as Responsive Accessibility Failures (RAFs), are difficult to identify using existing automated accessibility tools.

This paper presents an automated approach to detect reflow accessibility failures in responsive web pages. The proposed method models keyboard interactions by constructing interactive representations of user interfaces in both full-sized and reflowed layouts. Interactive elements are grouped into functional units using textual and structural similarity analysis. Functional equivalence across layouts is then evaluated to identify missing or inaccessible features under reflow conditions.

Experimental evaluation conducted on real-world web pages demonstrates that the proposed approach effectively identifies reflow-induced accessibility failures and achieves higher detection accuracy compared to existing tools. The results highlight the importance of functional-level accessibility analysis and support the development of more inclusive responsive web applications.

Keywords:- Responsive Web Design, Web Accessibility, Reflow, Keyboard Navigation, Assistive Technologies

1. INTRODUCTION

The web has become an essential platform for delivering information, services, and applications across diverse domains such as education, healthcare, finance, and governance. Ensuring accessibility in web applications is therefore a fundamental requirement rather than an optional feature. Web accessibility aims to provide equal access to information and functionality for all users, including individuals with visual, motor, cognitive, or auditory impairments. To support this goal, international standards such as the Web Content Accessibility Guidelines (WCAG) emphasize that web content must be perceivable, operable, understandable, and robust.

Responsive Web Design (RWD) has emerged as a widely adopted solution for accommodating the growing diversity of devices and screen sizes. Through techniques such as flexible layouts, media queries, and adaptive components, RWD enables a single web interface to adjust dynamically across desktops, tablets, and mobile devices. However, while RWD improves visual adaptability, it can unintentionally introduce accessibility barriers when not implemented carefully.

A critical challenge arises when interactive functionalities available in the full-sized layout become inaccessible in reflowed layouts. For users relying on keyboard navigation or assistive technologies, hidden menus, inaccessible buttons, and unreachable form controls can prevent successful interaction with the web application. These reflow-related accessibility issues are difficult to identify using conventional automated tools, which often focus on visual or syntactic violations rather than functional accessibility. This motivates the need for automated approaches that specifically target reflow accessibility failures.

2. METHODOLOGY

The proposed methodology focuses on identifying reflow accessibility failures by modeling and analyzing keyboard-based interactions within responsive web interfaces. The approach begins by rendering the web page in two distinct layouts: a full-sized viewport representing the standard desktop view and a reflowed viewport representing a reduced or magnified view consistent with accessibility evaluation standards.

In each layout, the system dynamically explores the user interface by simulating keyboard navigation. Interactive elements such as links, buttons, form controls, and custom interactive components are identified through runtime analysis of the Document Object Model (DOM). User Interface Interactive Models are constructed to capture navigation paths, focus transitions, and state changes triggered by keyboard actions.

To reason about functionality, interactive elements are grouped into functional units based on semantic similarity. This grouping considers multiple attributes, including element type, associated labels, textual content, and behavioral characteristics. By abstracting individual elements into functionalities, the approach enables meaningful comparison across layouts even when the visual structure or DOM hierarchy differs.
Once functional units are identified in both layouts, the system evaluates keyboard accessibility by analyzing reachability and actionability. Functionalities that are accessible in the full-sized layout but missing or inaccessible in the reflowed layout are classified as reflow accessibility failures. This functional-level comparison allows the detection of accessibility issues that are overlooked by traditional rule-based tools.

3. EXPERIMENTAL RESULTS

The proposed approach was implemented as an automated prototype and evaluated on a diverse set of real-world responsive web pages. The evaluation focused on three primary aspects: detection accuracy, execution efficiency, and the severity of identified accessibility failures. Ground truth was established through manual inspection following established accessibility evaluation procedures.
Experimental results demonstrate that the proposed approach effectively identifies a high proportion of reflow accessibility failures. Compared to existing accessibility testing tools, the approach achieves significantly higher recall and precision in detecting functionality-level accessibility issues. The evaluation also indicates that the approach operates within acceptable execution time, making it suitable for practical use during development and testing phases.

3.1 Experimental Setup

The proposed approach was implemented as an automated prototype capable of simulating keyboard-based interaction with responsive web pages. Each subject web page was evaluated under two configurations: a full-sized viewport representing standard desktop usage and a reflowed viewport representing reduced or magnified layouts commonly encountered by users of assistive technologies.

3.2 Evaluation Metrics

The effectiveness of the proposed approach was evaluated using standard performance metrics commonly employed in accessibility testing research. Precision was used to measure the proportion of correctly identified reflow accessibility failures among all reported issues, while recall measured the proportion of actual failures successfully detected by the system.

3.3 Detection Accuracy Analysis

Experimental results indicate that the proposed approach achieves high detection accuracy for reflow accessibility failures. A significant proportion of failures identified in the ground truth were correctly detected by the system, demonstrating strong recall performance. The approach also maintained high precision by minimizing false positives, which is critical for reducing developer overhead during remediation.

3.4 Runtime Performance Evaluation

The execution time of the proposed approach was evaluated across all subject web pages. Results show that the approach operates within acceptable time limits, even for pages with complex layouts and multiple interaction states. While dynamic interaction modeling introduces additional overhead compared to static analysis tools, the observed runtime remains practical for integration into accessibility testing workflows.
The detected failures were found to have substantial impact on usability, often preventing keyboard-only users from accessing critical navigation paths or core functionalities. These findings reinforce the importance of automated reflow accessibility testing as a complementary technique to existing accessibility evaluation tools.

4. CONCLUSION

Responsive Web Design has become an essential practice in modern web development; however, ensuring accessibility across reflowed layouts remains a significant challenge. This study addressed the critical problem of reflow accessibility failures, where essential functionalities available in full-sized web interfaces become inaccessible to keyboard-based users in reduced or reflowed viewports. Such failures disproportionately affect users who rely on assistive technologies and represent a major barrier to inclusive web access.
The paper presented an automated, functionality-aware approach for detecting reflow accessibility failures in responsive web interfaces. By dynamically modeling keyboard interactions and grouping interactive elements into functional units, the proposed methodology enables accurate comparison of accessibility across different layouts. Unlike traditional accessibility evaluation tools that focus primarily on static rules or visual presentation issues, the proposed approach emphasizes functional operability, which is central to real-world accessibility compliance.

Experimental evaluation on real-world web pages demonstrated that the approach is effective in identifying accessibility failures introduced by reflow, achieving high detection accuracy while maintaining practical execution performance. The results highlight that many critical accessibility issues remain undetected by existing tools, underscoring the necessity of interaction-driven and functionality-level analysis.

In conclusion, this work contributes toward improving inclusive web design by providing an automated solution capable of uncovering reflow-related accessibility barriers. By integrating such techniques into development and testing workflows, developers can proactively identify and resolve accessibility issues, ultimately ensuring equal access to web content and services for all users, regardless of device or ability.

5. REFERENCES

[1] World Wide Web Consortium (W3C), *Web Content Accessibility Guidelines (WCAG) 2.1*, W3C Recommendation, 2018.
[2] World Wide Web Consortium (W3C), *Understanding Success Criterion 1.4.10: Reflow*, W3C, 2018.
[3] S. Harper and Y. Yesilada, *Web Accessibility: A Foundation for Research*, Springer, 2008.
[4] A. Cooper, R. Reimann, and D. Cronin, *About Face: The Essentials of Interaction Design*, Wiley, 2014.
[5] J. Lazar, J. H. Feng, and H. Hochheiser, *Research Methods in Human-Computer Interaction*, Morgan Kaufmann, 2017.
[6] A. Sears and J. A. Jacko, *Human-Computer Interaction: Development Process*, CRC Press, 2009.
[7] M. R. C. A. Santos, F. P. M. Silva, and A. R. Rocha, "Accessibility evaluation of responsive web applications," *International Journal of Web Engineering*, vol. 12, no. 3, pp. 145–160, 2020.
[8] E. Marcotte, "Responsive web design," *A List Apart*, no. 306, 2010.
[9] G. Brajnik, "Automatic web accessibility evaluation: Where are we?," *Universal Access in the Information Society*, vol. 7, no. 1, pp. 45–59, 2008.
[10] S. Vigo, J. Brown, and V. Conway, "Benchmarking web accessibility evaluation tools," *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility*, pp. 1–10, 2013.
[11] R. K. Bellamy et al., "Improving web accessibility for keyboard-only users," *IBM Journal of Research and Development*, vol. 57, no. 3, pp. 1–12, 2013.
[12] T. Grossman, G. Fitzmaurice, and R. Attar, "A survey of software learnability," *ACM Computing Surveys*, vol. 44, no. 3, pp. 1–36, 2012.
[13] P. Acosta-Vargas, S. Luján-Mora, and L. Salvador-Ullauri, "Web accessibility evaluation methods: A systematic review," *IEEE Access*, vol. 7, pp. 139000–139015, 2019.
[14] H. Petrie and N. Bevan, "The evaluation of accessibility, usability, and user experience," *The Universal Access Handbook*, CRC Press, 2009.
[15] A. Holzinger, "Usability engineering methods for software developers," *Communications of the ACM*, vol. 48, no. 1, pp. 71–74, 2005.
[16] M. B. Rosson and J. M. Carroll, *Usability Engineering: Scenario-Based Development of Human-Computer Interaction*, Morgan Kaufmann, 2002.
[17] ISO/IEC 40500:2012, *Information Technology — W3C Web Content Accessibility Guidelines (WCAG) 2.0*, ISO Standard.
[18] D. Theofanos and B. Redish, "Bridging the gap: Between accessibility and usability," *Interactions*, vol. 10, no. 6, pp. 36–51, 2003.