Accessibility Metatesting

Comparing Nine Testing Tools

Jonathan Robert Pool
jonathan.pool@cvshealth.com
CVS Health
Woonsocket, Rhode Island, USA

ABSTRACT

Automated web accessibility testing tools have been found complementary. The implication: To catch as many issues as possible, use multiple tools. Doing this efficiently entails integration costs. Is there a small set of tools that, together, make additional tools redundant? I approach this problem by comparing nine comprehensive accessibility testing tools that are amenable to integration: alfa, axe-core, Continuum, Equal Access, HTML CodeSniffer, Nu Html Checker, QualWeb, Tenon, and WAVE. I tested 121 web pages of interest to CVS Health with these tools. Each tool only fractionally duplicated any other tool. Each discovered numerous issue instances missed by all the others. Thus, testing with all nine tools was substantially more informative than testing with any subset.

CCS CONCEPTS

• **Software and its engineering** → *Empirical software validation*; **Software testing and debugging**; • **Human-centered computing** → *Empirical studies in accessibility*; **Accessibility design and evaluation methods**; **Accessibility technologies**; **Accessibility systems and tools**.

KEYWORDS

web accessibility, accessibility testing, metatesting, test automation, test efficiency

ACM Reference Format:
Jonathan Robert Pool. 2023. Accessibility Metatesting: Comparing Nine Testing Tools. In *20th International Web for All Conference (W4A '23), April 30-May 1, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3587281.3587282

1 INTRODUCTION

Organizations that test their own, their competitors', their suppliers', and other web assets for accessibility [19] can handle part of the testing workload with automated and semi-automated tools. The World Wide Web Consortium (W3C) lists 167 such tools [21]. Some are designed for the fully automated and comprehensive discovery of accessibility issues. They include web services, APIs, browser extensions, and installable software.

W4A '23, April 30-May 1, 2023, Austin, TX, USA

ACM ISBN 979-8-4007-0748-3/23/04…$15.00
https://doi.org/10.1145/3587281.3587282

Comparisons of such tools have found them substantially complementary: Issues discovered by one are often overlooked by another. This has led to recommendations to use multiple tools rather than relying on only one [1][9].

The more tools one uses, the greater the cost of integrating them into a regime of automated accessibility testing. This motivates the question, how many, and which, tools to deploy. The most obvious candidates are free or nearly free tools that conform to standard protocols, can be controlled programmatically, and aim to test accessibility comprehensively.

I explore the tool-selection problem here by studying the issues reported by nine accessibility testing tools amenable to integration.

2 RELATED WORK

Abduganiev [1] compared eight accessibility testing tools on fifty-two web pages from Tajikistan and Austria, and Pădure and Pribeanu [9] compared six accessibility testing tools on the websites of six municipal governments in Romania. Both reported enough complementarity to justify using multiple tools.

Ara and Sik-Lanyi [2] used four tools to test the accessibility of twenty COVID-19 vaccination websites of governments of mostly European nation states. They found large inter-tool differences in reported error counts. For example, the Mauve tool found 12 errors on the Dutch site and 16 errors on the German site, but the Web Accessibility tool found 104 errors on the Dutch site and 1 error on the German site.

Burkard, Zimmermann, and Schwarzer [3] evaluated four commercial accessibility-monitoring tools. The one that discovered the most issues did not render redundant the one that discovered the fewest. The latter still discovered violations of six accessibility standards that the former did not discover [4].

Silva, Oliveira, Mateus, Costa, and Freire [12] reviewed four studies and tabulated the accessibility issues reported by the tools they had employed. The analysis enables a comparison of only two tools: SortSite and WAVE. One issue was reported by both; one was reported by only SortSite; and nine were reported by only WAVE. Thus, the tools were almost totally complementary.

The reviewed research supports the generalization that purportedly comprehensive accessibility testing tools differ substantially in the issues they report.

3 TOOLS

I classify tools as amenable to integration into a multi-tool regime of automated testing if they are currently maintained, free or nearly free, and either usable as REST APIs or installable as NPM packages. By this classification, of the fourteen tools used in the studies cited above, two, and derivatives of two more, are amenable to integration.

Table 1: Accessibility Testing Tools

Code	Name	Creator	Tests
alfa	alfa [13]	Siteimprove	103
axe	axe-core [5]	Deque	138
continuum	Continuum [8]	Level Access	267
htmlcs	HTML CodeSniffer [14]	Squiz	98
ibm	Equal Access [6]	IBM	163
nuVal	Nu Html Checker [20]	W3C	147
qualWeb	QualWeb [7]	Universidade da Lisboa	121
tenon	Tenon [15]	Tenon.io	180
wave	WAVE [16]	WebAIM	110
Total			1,327

Table 2: Counts of Issue Instances Reported

Tool	Count
qualWeb	23,715
axe	12,364
tenon	8,328
nuVal	6,986
htmlcs	6,329
ibm	6,242
wave	6,095
alfa	5,605
continuum	3,089
Total	78,753

I have chosen these four and five additional qualifying tools for comparison here. The tools I compare are listed in Table 1. They all seek to measure conformance to industry standards [18] and best practices for accessibility, so it is reasonable to speculate that some of them might make some others of them redundant.

The nine tools compared here should not be expected to remain viable candidates, or the only viable candidates, because, as Abduganiev [1] notes, tools rapidly appear and disappear. In fact, one of the tools I compare (Tenon) became unavailable for new subscribers after I conducted this research.

4 METHODS

I classified the tools' 1,327 tests into 245 "issues" (defects and suspected defects), such that any reports of the same issue are arguably duplicative. For example, the "labelClash" issue is a form control improperly having multiple labels. Defects reported as definitive are classified as different issues from mere suspicions or warnings. The supplementary materials include the complete issue classification.

One could analyze tool complementarity *a priori* on the basis of such a classification to determine which tools can discover which issues. For example, 3 of the 9 tools (axe-core, IBM Equal Access, and WAVE) have tests classified as "labelClash".

However, an *a priori* analysis would lead to decisions based on *purported* tool capabilities, which might only later be found inaccurate. Instead, I approach the problem empirically by analyzing the issues that the tools actually report.

Using the case of CVS Health, I assembled a judgmental sample [17] of 140 web pages that an enterprise might monitor for accessibility—internal home-built, internal vendor-produced, external own, external supplier, and external competitor pages. I then removed from the sample any publicly unreachable pages and any pages requiring pre-test interaction (such as logging in), to allow externally hosted tools without scripted actions to test the whole sample. This left a sample of 121 pages.

I tested all 121 pages with all nine tools. This permitted an empirical tabulation of duplication and complementarity among the tools. With this empirical approach, a tool's presumed value is based on the issue instances that it reports but other tools do not.

This method, although empirical, is still naïve:

- It trusts the claims of each tool as to the issue instances it has discovered.
- It assumes that, if on some page tool A reports m instances of issue C and tool B reports n instances of issue C, where $n > m$, then the instances reported by A are a subset of the instances reported by B.

Thus, in the analysis below, references to issues discovered are, strictly speaking, references to *claims* of issues discovered.

My trust in the claims of tools contrasts with prior studies [1][3] that employed human testers to classify automatically reported issues as correct ("true positives") or incorrect ("false positives").

There are reasons for such trust. False positivity is, arguably, a judgment, not a fact. Moreover, litigators reportedly use automated tests to identify inaccessible websites [10][11]; thus, treating automated failure reports as presumptively correct, thereby avoiding not only inaccessibility but the *appearance* of inaccessibility, can mitigate the risk of claims and disputes.

5 RESULTS

5.1 Totals

The tools differed substantially in how many issue instances they reported, with the most prolific tool reporting about eight times as many instances as the least prolific tool, as seen in Table 2.

This result could tempt one to conclude that the QualWeb tool does everything the other tools do and more, making it possible to use that tool alone without sacrificing any information, but that conclusion would not survive further analysis, as shown below.

5.2 Pairs

If one tool made the others redundant, then, if paired with each other tool, it would, on every tested page, report at least as many instances of each issue as the other tool reports.

In fact, however, with each of the 72 pairs of tools, each tool found, on one or more pages, more instances of some issues than the other tool found. For example, the alfa tool found 4,652 instances that the Continuum tool missed, but Continuum found 2,136 instances that alfa missed. Thus, for any two tools, testing with both revealed more instances than testing with either tool alone.

The pairwise percentage increases of issue instances from testing with two tools instead of one are shown in Table 3. For example, the "48" in the bottom row means that testing my sample with both WAVE and Continuum Community Edition produced reports of

Table 3: Pairwise Increase in Issue Instance Count (%)

1st	alf	axe	con	2nd htm	ibm	nuV	qW	ten	wav
alfa	-	193	38	105	96	123	402	143	100
axe	33	-	16	43	38	54	188	63	39
continuum	151	366	-	169	169	218	767	265	191
htmlcs	81	179	31	-	86	107	370	128	76
ibm	76	174	33	88	-	104	379	130	88
nuVal	79	173	40	87	82	-	339	119	85
qualWeb	19	50	13	25	26	29	-	33	24
tenon	64	142	35	74	73	84	279	-	61
wave	84	183	48	83	92	112	384	120	-

Table 4: Pairwise Increase in Issue Discovery

1st	alf	axe	con	2nd htm	ibm	nuV	qW	ten	wav
alfa	-	47	34	41	49	46	22	20	42
axe	25	-	25	40	43	40	21	20	42
continuum	34	46	-	41	46	42	25	21	44
htmlcs	31	39	28	-	45	45	21	19	39
ibm	29	40	26	36	-	43	23	20	40
nuVal	34	48	31	36	46	-	26	21	43
qualWeb	36	48	35	41	49	51	-	19	39
tenon	36	50	34	42	50	50	24	-	42
wave	33	41	28	31	41	46	22	15	-

48% more issue instances than testing with only WAVE. Adding a second tool increased the count by anywhere from 13% to 767%.

One might care how many additional issues a tool discovers, regardless of the instance count. That information, for this sample, is reported in Table 4. Thus, adding a second tool added instances of anywhere from 15 issues (if Tenon was added to WAVE) to 51 issues (if Nu Html Checker was added to QualWeb).

5.3 Issues

Testing tools, to some extent, specialize in issues. If an organization cares about particular accessibility issues, it might be practical to test mainly or only with tools proficient in those issues.

The most evident specializations included:

- alfa: font and line sizing; skip-to-content links
- axe-core: color contrast; landmark placement
- Continuum: landmark purpose; placeholder versus label
- HTML CodeSniffer: heading levels; semantic use of elements
- Equal Access: landmark coverage; landmark purpose
- Nu Html Checker: control placement; attribute validity
- QualWeb: video alternatives; focus indication
- Tenon: horizontal scrolling; link purpose
- WAVE: label clarity; link purpose

If, however, an organization cares about all accessibility issues, selecting a subset of tools is problematic. In my sample, every tool had at least 10 issues that it found more instances of than did any other tool. For example, five of the tools found instances of a bad iframe title, but HTML CodeSniffer found 108 such instances, while no other tool found more than 27. The supplementary materials include a complete list of these specializations.

Table 5: Counts of issues Reported by Only One Tool

Tool	Count
nuVal	26
htmlcs	15
ibm	15
axe	12
wave	12
continuum	9
alfa	8
tenon	8
qualWeb	7
Total	112

Moreover, every tool had at least 7 issues that *only* that tool found instances of, as shown in Table 5. For example, only WAVE found instances of pages entirely missing landmarks, and only axe-core found instances of invisible form-control labels. The supplementary materials include a complete list of these sole-source issues.

In some cases, the failure of a tool to discover instances of an issue on a page was due to the inability of that tool to test that page at all. Of 1,089 attempts to test a page with a tool, 49 (4%) failed. A tool that successfully tests a page and does not find some issue has the same practical effect as a tool that fails to test the page: That tool does not discover that issue, but another tool might.

5.4 Duplication

While tools complemented each other, they also duplicated each other. The number of tools discovering issue instances ranged from 1 to 8. For example, 8 tools discovered links without accessible names. Even so, the duplication was imperfect. For example, WAVE found 240 such links, while the Nu Html Checker found only 2.

6 CONCLUSION

Although all the tools except Nu Html Checker claim to be comprehensive web-accessibility testers, with my sample of web pages each tool only fractionally duplicated any other tool. Each discovered numerous instances missed by the others, and each was the only one to discover instances of some issues. For this sample, testing with all nine tools was substantially more informative than testing with any smaller set or with only a single tool.

No single accessibility testing tool can be trusted to discover all automatically detectable defects. There is reason to expect more discovery from testing with multiple tools.

Routinely and efficiently testing for accessibility with as many as nine tools, even if free or inexpensive, involves effort to:

- integrate distinct tool invocation methods
- ensure test isolation
- integrate distinct reporting formats of tools
- integrate distinct tool severity and certainty classifications
- integrate distinct methods used by tools to identify instance locations

- integrate distinct methods allowed by tools for pre-test browser actions
- reconcile granularity differences between the issue classifications of tools
- determine which issues should be considered identical
- assign levels of trust to tools and their tests [22]
- adjust defect weights in page-scoring algorithms to compensate for tool duplicativity
- report findings on one issue together, not in separate tool reports
- keep track of the revisions of each tool
- handle tool disagreements
- integrate multiple tools with internally created tests and standards

In judging whether the benefits justify this effort, one can consider that the investment is, in part, nonrepeating, amortizable over multiple uses, and sharable by multiple organizations. Moreover, testing with an ensemble of tools can influence tool quality, by facilitating direct comparisons among tool behaviors, helping tool users discover and report tool deficiencies, and motivating the development of new, nonduplicative accessibility tests.

A DISCLAIMER

My opinions expressed herein are my own views and do not necessarily reflect the views of CVS Health, its affiliates, or any of my colleagues at CVS Health or its affiliates.

B APPENDIX: SUPPLEMENTARY DOCUMENTS

The following documents supplement this paper.

B.1 Issue Classification

The "issueClassification.json" file defines the issue classification used in this investigation. For each issue, the most applicable Web Content Accessibility Guidelines 2.1 [18] Success Criterion, Guideline, or Principle and the tests detecting the issue are given. For each test, the tool's own test ID, whether that ID is "variable" (i.e. is a regular expression to be matched rather than a string), and a paraphrase of the tool's description of the issue are given.

B.2 Sole-Source Issues

The "onlies.json" file documents the issues discovered by only one tool each. The count of issue instances found, an ID of the issue in the issue classification, and a paraphrase of the issue are given.

B.3 Tool Issue Tabulation

The "issues.json" file documents, in its "issues" object, the number of instances of each issue reported by each tool. In its "most" object, it also documents the issues that each tool reported more instances of than any other tool. For example, the fact that "legendMissing" is in the "most.ibm.issues" array means that the "ibm" tool reported more instances of the "legendMissing" issue than any other tool.

REFERENCES

[1] Siddikjon Gaibullojonovich Abduganiev. 2017. *Towards Automated Web Accessibility Evaluation: A Comparative Study.* Retrieved 2023-01-12 from https://www.mecs-press.org/ijitcs/ijitcs-v9-n9/IJITCS-V9-N9-3.pdf

[2] Jinat Ara and Cecilia Sik-Lanyi. 2022. *Investigation of COVID-19 Vaccine Information Websites across Europe and Asia Using Automated Accessibility Protocols.* Retrieved 2023-03-04 from https://www.mdpi.com/1660-4601/19/5/2867

[3] Andreas Burkard, Gottfried Zimmermann, and Bettina Schwarzer. 2021. *Monitoring Systems for Checking Websites on Accessibility.* Retrieved 2023-03-04 from https://www.frontiersin.org/articles/10.3389/fcomp.2021.628770/full

[4] Andreas Burkard, Gottfried Zimmermann, and Bettina Schwarzer. 2021. *Monitoring Systems for Checking Websites on Accessibility: Supplementary Materials.* Retrieved 2023-03-04 from https://www.frontiersin.org/articles/10.3389/fcomp.2021.628770/full#supplementary-material

[5] Deque. 2023. *axe-core.* Retrieved 2023-01-11 from https://www.npmjs.com/package/axe-core

[6] IBM. 2023. *Equal Access Accessibility Checker.* Retrieved 2023-01-11 from https://www.npmjs.com/package/accessibility-checker

[7] Informática, Universidade da Lisboa. 2023. *QualWeb.* Retrieved 2023-03-11 from https://www.npmjs.com/package/@qualweb/core

[8] Level Access. 2023. *Continuum Community Edition.* Retrieved 2023-01-11 from https://www.webaccessibility.com/tools/

[9] Marian Pădure and Costin Pribeanu. 2020. *Comparing Six Free Accessibility Evaluation Tools.* Retrieved 2023-03-03 from https://protect-us.mimecast.com/s/7bi1CG6o9qf00QjmET7sgkW?domain=researchgate.net

[10] Kris Rivenburgh. 2019. *Lawsuit Website Accessibility is Different; Automated Checkers Give Rise to New Issue Post Remediation.* Retrieved 2023-03-14 from https://krisrivenburgh.medium.com/lawsuit-website-accessibility-is-different-automated-checkers-give-rise-to-new-issue-post-f23d6bef188f

[11] Adrian Roselli. 2022. *#accessiBe Will Get You Sued.* Retrieved 2023-03-14 from https://adrianroselli.com/2020/06/accessibe-will-get-you-sued.html#Lawsuits

[12] Carlos Alberto Silva, Arthur F. B. A. de Oliveira, Delvani Matônio Mateus, Heitor Augustus Xavier Costa, and André Pimenta Freire. 2019. *Types of Problems Encountered by Automated Tool Accessibility Assessments, Expert Inspections and User Testing: A Systematic Literature Mapping.* Retrieved 2023-03-13 from https://doi.org/10.1145/3357155.3358479

[13] Siteimprove. 2023. *alfa.* Retrieved 2023-01-11 from https://alfa.siteimprove.com/

[14] Squiz. 2023. *HTML CodeSniffer.* Retrieved 2023-01-11 from https://www.npmjs.com/package/html_codesniffer

[15] Tenon.io. 2023. *Tenon.* Retrieved 2023-01-11 from https://tenon.io/documentation/apiv2.php

[16] WebAIM. 2023. *WAVE.* Retrieved 2023-01-11 from https://wave.webaim.org/api/

[17] Linda Westfall. 2009. *Sampling Methods.* Retrieved 2023-03-04 from http://kajabi-storefronts-production.s3.amazonaws.com/sites/69255/themes/2149842111/downloads/IHrhnXpIShOUIuvRhouL_Sampling_Methods.pdf

[18] World Wide Web Consortium. 2018. *Web Content Accessibility Guidelines (WCAG) 2.1.* Retrieved 2023-01-13 from https://www.w3.org/TR/WCAG21/

[19] World Wide Web Consortium. 2023. *Introduction to Web Accessibility.* Retrieved 2023-01-12 from https://www.w3.org/WAI/fundamentals/accessibility-intro/

[20] World Wide Web Consortium. 2023. *Nu Html Checker.* Retrieved 2023-01-11 from https://github.com/validator/validator

[21] World Wide Web Consortium. 2023. *Web Accessibility Evaluation Tools List.* Retrieved 2023-01-11 from https://www.w3.org/WAI/ER/tools/

[22] Ulaş Yüksel, Hasan Sözer, and Murat Şensoy. 2014. *Trust-based Fusion of Classifiers for Static Code Analysis.* Retrieved 2023-01-12 from http://confcats_isif.s3.amazonaws.com/web-files/event/proceedings/html/2014Proceedings/papers/fusion2014_submission_118/paper118.pdf

Received 17 January 2023; accepted 28 February 2023; revised 14 March 2023