Large Scale Automatic Web Accessibility Validation

Large Scale Automatic ...

Nicola Iannuzi, Marco Manca, Fabio Paternò, Carmen Santoro

CNR-ISTI, HIIS Laboratory, {nicola.iannuzzi, marco.manca, fabio.paterno, carmen.santoro}@isti.cnr.it

Digital accessibility is considered an important aspect to allow all people, including those with permanent or temporary disabilities, to access the continuously increasing number of digital services. This raises the need for tools able to provide support for monitoring the level of accessibility of a large number of websites in order to understand their actual level of accessibility, and identify the areas that need more interventions for their improvement. We present how we have extended a tool for accessibility validation for this purpose, and the results that we obtained in the validation of about 2.7 million Web pages of Italian public administration Web sites.

CCS CONCEPTS • Human-centered computing → Accessibility systems and tools;

Additional Keywords and Phrases: Accessibility, Automatic Validation Tools, Large-scale validations

ACM Reference Format:

First Author's Name, Initials, and Last Name, Second Author's Name, Initials, and Last Name, and Third Author's Name, Initials, and Last Name. 2018. The Title of the Paper: ACM Conference Proceedings Manuscript Submission Template: This is the subtitle of the paper, this document both explains and embodies the submission format for authors using Word. In Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY. ACM, New York, NY, USA, 10 pages. NOTE: This block will be automatically generated when manuscripts are processed after acceptance.

1 INTRODUCTION

Digital accessibility for all people, including those with permanent or transient disabilities, is becoming more and more important for the continuous need to access digital services in our daily lives. In order to guarantee this possibility many countries have national legislations, and specific directives have been promoted in Europe. In particular, the WAD directive [EU 2016] has indicated that all European countries should also monitor the state of the accessibility of web and mobile applications. However, despite the increasing attention at a legislative, academic, and social level, many public websites are still not able to meet the minimum level of accessibility requirements [Gaggi et al., 2022].

In the meantime, the W3C Web Content Accessibility Guideline (WCAG) have evolved in order to address the evolution of the technologies for implementing interactive applications and better address the needs of the various possible disabilities. The WCAG 2.1 indicates 78 success criteria, and many more techniques. In addition, the way how developers implement their Web sites is evolving as well, with the increasing use of newer versions of frameworks for developing dynamic sites such as React and Angular.

All such factors have increased the complexity of the validation of Web applications, which consequently requires considerable effort and can be rather tedious if performed manually. For such reasons, interest in automatic support of accessibility validation has increased, even if it is well-known that not all the guidelines for accessibility can be automatically performed [Power et al., 2012], and direct user feedback is still necessary. Indeed, several tools for automatic validation have been put forward. As of May 2023, the W3C Web Accessibility Evaluation Tools list[1] contains 167 tools for accessibility validation. However, accessibility evaluation is an area continuously evolving and several tools have not been able to cope with such evolutions, thus they have become obsolete because they address old versions of the accessibility guidelines, or they are not able to address modern dynamic websites, or they are limited in terms of scalability of the number of pages that they are able to validate.

In particular, the continuously increasing digitalization of contents and services and the request from public authorities to monitor the accessibility state of large numbers of websites have stimulated the need for tools able to address large-scale validations. Unfortunately, this is a challenge that has received limited attention so far and requires more reflection on how accessibility validators have to evolve in order to address them.

In this paper, we aim to contribute to filling this gap by discussing how a tool for automatic Web accessibility validation (MAUVE++) (Iannuzzi et al., 2022) has been extended in several aspects to address large-scale validations and the results that it has provided when applied to the Web sites of the Italian public administrations, which resulted in an analysis of 2.7 million of web pages. Thus, we discuss the type of validation and results required by a large-scale validation for monitoring purposes, and the technical aspects that need to be addressed to support the scalability and generality of the approach.

2 RELATED WORK

Evaluating Web accessibility requires checking and monitoring many details across the pages of a website. Even though accessibility validation is a process that cannot be fully automated (Vigo et al., 2013), to simplify the monitoring, analysis, detection, and correction of website accessibility problems, many automatic and semi-automatic tools have been proposed over the years (e.g. Beirekdar et al., 2005; Schiavone and Paternò 2015) to help in this regard.

However, one well-known issue when using automatic validators for checking web accessibility is that such tools can radically differ according to various aspects (Abascal et al., 2019), from the coverage of accessibility guidelines, to how tools interpret and to what extent they are able to support the considered guidelines, to the way such tools present the results including errors and warnings (which require manual intervention to be evaluated). Moreover, validators can even provide different results when evaluating the same Web content (Abduganiev, 2017) also due to the ambiguity of guidelines themselves (Pelzetter 2021).

One main issue associated with the above-mentioned differences among accessibility tools is that they can be perceived in different ways by users, are sometimes misinterpreted, and can generate misunderstandings, also because they sometimes are not clear about what they actually validate: thus, there is a need to make such tools more transparent for their users, as also highlighted in (Manca et al., 2022). Analysis of requirements that should characterise accessibility evaluations (Yesilada et al., 2019) and a new generation of tools for supporting accessibility validation obtained by involving several stakeholders (Paternò et al., 2020) have been reported. Several studies on accessibility tools have been carried out as well. For example, a detailed study on the results of automatic Web accessibility evaluation provided by several tools is reported in (Abduganiev, 2017), which considered support for just WCAG 2.0 guidelines and analysed eight popular and free online automated Web accessibility evaluation tools, finding significant differences among them in terms of various aspects

[1] https://www.w3.org/WAI/ER/tools/

(coverage, completeness, correctness, validity, efficiency and capacity). Padure and Pribeanu (2019) compared five automatic tools for assessing accessibility, showing that the combined use of two of the considered tools would increase the completeness and reliability of the assessment. More recently, Burkard et al. (2021) compared four commercial monitoring accessibility tools (SiteImprove, PopeTech, aXe Monitoring and ARC Monitoring), by evaluating them on only five Web pages according to criteria such as coverage of the Web pages, success criteria, completeness, correctness, support for localisation of errors, and manual checks. However, their analysis did not focus on aspects such as monitoring and support for dynamic sites.

In spite of the limitations associated with automated accessibility evaluation tools, their use is really essential when it comes to evaluating (quite regularly) a high number of web pages, as it should be done in accordance with the European WAD Directive for accessibility of websites and mobile applications of public sector bodies. In reply to the issue of that directive, some Member States across Europe started to implement large-scale monitoring of the adoption of accessibility guidelines, whose outcomes are aimed at stimulating further in-depth evaluation[2].

In this regard, Martins and Duarte (2022) recently considered web accessibility metrics, i.e. quantitative indicators obtained through specific formulas, which are applied using data provided by accessibility evaluations, and are aimed at synthesising the accessibility level of a web resource into a specific value. In particular, for their analysis, they considered and compared eleven web accessibility metrics, computed using the QualWeb tool[3], and a sample of around three million web pages, taken from an open corpus of web data. They applied not the WCAG guidelines but the ACT Rules (https:// act- rules. github. io/ rules/) for the evaluation. By computing all the metrics over this sample of web pages, the authors aimed to understand if these metrics could correlate with each other, in particular, if there are groups of metrics that offer similar results. However, such metrics are often too technical for the accessibility stakeholders that have difficulties understanding their meaning. Thus, in our study, which analysed a number of web pages of similar size, we also expressed the results of the evaluation using two easy-to-understand metrics (Broccia et al., 2020) to facilitate comparison across sites. Thus, differently from those authors, the goal of our study is not to analyse the suitability of accessibility metrics, but we only used them to provide useful synthetic information on the results obtained by the accessibility monitoring process on the considered web pages.

Even before the advent of the WAD, researchers started to analyse the state of accessibility of websites at a medium-large scale. Indeed, a first exploration of how to support monitoring of Web sites accessibility at a geopolitical level was already discussed in (Mirri et al., 2011). However, the tool presented in that paper provided only some limited representations (in a tabular form) of the accessibility levels, and it considered older WCAG (e.g. 2.0) versions. More recent examples of early investigations of accessibility evaluations done at geo-political levels have been reported for countries such as Romania (Pribeanu, 2019) and Norway (Inal et al., 2022). In particular, Inal et al. (2022), beyond analysing Norway's situation, conducted an analysis (also discussing similar medium/large-sized scale studies done in countries such as Poland, Slovenia, and Bulgaria), showing a weak negative correlation between the population of municipalities and the number of success criteria violated. In Italy, some early preliminary studies about the situation of web accessibility of public administrations have been carried out as well. Beyond the already mentioned study of Mirri et al. (2011) which reported results of evaluating about 4000 pages of Emilia-Romagna region, Gambino et al. (2014) analysed the official web pages of Italian provinces' and regions' chief towns to check their compliance to the 22 technical requirements defined by the Stanca Act (the Italian main reference point for accessibility in Italy). A sample of 976 web

[2] https://digital-strategy.ec.europa.eu/en/library/web-accessibility-directive-monitoring-reports
[3] http://qualweb.di.fc.ul.pt/evaluator/

pages belonging to the websites of the Italian chief towns was submitted to the Achecker tool[4], and several accessibility and syntax errors were found which involved at least 10 of the 22 technical requirements stated in the Stanca Act, thus showing that the Italian institutional websites considered in that work were not accessible. Some years later, Barricelli et al. (2018) presented a survey regarding the accessibility of Italian municipal websites, with the goal of checking their compliance with the Stanca Act (aligned with WCAG 2.0). In particular, they analysed the home pages of 8057 Italian municipalities with Achecker, showing that most of them were not accessible. In the same study, the authors also tried to figure out the efforts needed to fix the accessibility problems that affected these websites, by classifying the problems according to the kind of intervention needed to solve them.

Overall, it emerges that a large-scale validation of Italian Web sites is still missing since the first attempts that considered a limited number of pages, applying older versions of the WCAG guidelines, and using tools that were not able to support validation of dynamic content, such as Achecker.

3 HOW THE LARGE-SCALE VALIDATION PROCESS HAS BEEN ADDRESSED

To support large-scale validation, we have extended the MAUVE++ tool[5]. When we started this study, it was a tool aiming at identifying accessibility errors on websites in order to facilitate their correction. For this reason, it focused on presenting results in terms of violated techniques (which is the most detailed level), and it was able to validate only small groups of pages associated with a Web site. Thus, we had to face three different challenges to perform a large-scale validation: to report the validation results in a more abstract manner and higher level granularity to facilitate their interpretation from a monitoring perspective, to update the crawling and the validation process in order to be able to scale-up to large numbers of pages and sites, and to support validation of Web sites implemented with modern technologies.

3.1 Provide More Abstract and Coarse-Level Granularity in Validation Results

The international accessibility guidelines (W3C WCAG) are structured into four levels: principles, guidelines, success criteria, and techniques. Originally MAUVE++ was designed to provide results at the most detailed level (the techniques) to facilitate the identification and corrections of errors. From the discussion with various stakeholders (accessibility evaluators, monitoring authorities) it emerged that this type of analysis is too detailed and fragmented for the purposes of monitoring activities, and therefore there was a need to present the results also in terms of a higher abstraction level indicated by the WCAG (the success criteria level). However, the relations between success criteria and techniques are not trivial, thus it was necessary to further deepen their associations in the validation process, and how to address them to provide meaningful results.

Generally speaking, there is a many-to-many relationship between success criteria and techniques since usually one success criterion is associated with several techniques, and one technique can be relevant for several success criteria. In addition, the techniques suggested by the W3C for each success criterion are of three types: Failure; Sufficient; Advisory. Techniques marked as Advisory are suggested methods for improving accessibility, but these do not determine compliance or non-compliance with the success criterion for which they are suggested. Sufficient techniques are methods known to reliably address particular accessibility barriers: thus, when they are applied, they ensure that the associated success criterion is satisfied as well. However, the opposite is not true: if an element does not correctly implement a sufficient technique, this does not mean that its content does not satisfy the success criterion under analysis. An element that does

[4] https://achecker.achecks.ca/checker/index.php
[5] https://mauve.isti.cnr.it/

not correctly implement Failure techniques results in failure to meet the associated success criterion. The opposite is not true: if an element passes one of the failure technique checks, this does not mean that its content satisfies the success criterion under analysis.

One further aspect to be aware of is that in general, the technique validations can have three results: Success, when the test is passed; Failure, when the test has a negative outcome; Cannot Tell when the system declares that it cannot offer a certain answer and suggests further human manual investigation. MAUVE++ validates a web page considering the techniques regardless of their type (Failure, Sufficient, Advisory). For each element of the web page, the tool calculates the result of the success criterion based on the results of the evaluated associated techniques.

Applying strictly the definitions of the techniques types, it was noticed that many of the occurrences of the success criteria results ended up in the "Cannot Tell" category, a result with which the system declares that it cannot provide a certain answer. This occurred for example when a failure technique passes, since this case cannot imply that the corresponding success criterion passes; or when a sufficient technique fails, as this cannot generate the failure of the success criterion to which it refers. Thus, in both situations, the resulting outcome is a “Cannot Tell”. To overcome this problem and considering that the W3C indicates "although techniques are useful for evaluating the content, evaluations must go beyond simply verifying sufficient technique tests to evaluate content compliance with WCAG success criteria." (https://www.w3.org/WAI/WCAG21/Understanding/understanding-techniques#testing-techniques), it was decided to provide more meaningful results by including in the "Cannot Tell" category only those occurrences generated by techniques with a "Cannot Tell" result, and not those corresponding to the passing of a failure technique or the failure of a sufficient technique.

The newly developed evaluation algorithm is able to manage cases in which a logical relationship applies to both groups of techniques and techniques within each group. In fact, for some success criteria, the documentation suggests groups of techniques that must be validated together to provide a result on the criterion. For example, for success criterion 4.1.1, the documentation specifies the following sufficient techniques (as a group), expressed with these relationships: G134 **OR** G192 **OR** H88 **OR** [(H74 **AND** H93 **AND** H94) **OR** H75].

To describe the relationships between techniques and success criteria, the XML-based language used by MAUVE++ (LWGD: Language for Web Guideline Definition) to drive the validation process has been modified to better address the relationships between success criteria and techniques. In particular, we added the possibility to specify the techniques that MAUVE++ uses to analyse the success criterion. This description includes the type of techniques (Failure, Sufficient, Advisory), the relationships between groups of techniques and the relationships between techniques within groups. Furthermore, now it is also specified whether the technique is implemented or not in MAUVE++ to ensure the accuracy of the output. This allows reporting the need for manual intervention when the tool is not able to verify one technique in a group of techniques that must be validated together.

These definitions are interpreted and applied by the software at the end of the validation of each success criterion, thus generating the result based on the assessments of the various techniques associated with it: the Java object generated for each success criterion reports the results of all techniques associated with it, and the relative XPaths of the elements on which the techniques checks were applied. For each technique, two lists of XPaths are generated, an *Error List* containing the XPaths of all the elements that did not pass the check, and a *Pass List* containing the XPaths of those that passed the check. Thanks to such data and to the information associated with the relationships between the techniques associated with each Success Criterion (as identifiable in the guideline definition file), it is possible to enumerate the occurrences of Success, Failures and Cannot Tell for each criterion analysed, based on the number of HTML elements that passed or failed the checks.

The algorithm analyses the techniques associated with each criterion taking into account their relations, since in some cases they are grouped through logical operators. The analysis starts from the basic elements and then considers the groups of techniques and their logical relations to identify the final results.

After having analysed the web page, the output produced by the algorithm for each SC consists of three lists: error, pass and cannot cell, in which the XPaths of the HTML elements are inserted based on the results obtained. These lists have no common elements, as an element can exclusively generate only one result per success criterion. The size of these lists represents the number of occurrences of the three results for a success criterion.

With the tool used in our study, two metrics are available. The *Accessibility Percentage* provides info about the number of distinct elements successfully evaluated out of the total number of techniques for which the tool was able to make a successful or unsuccessful evaluation, and the *Evaluation Completeness*, namely the number of distinct validation elements for which the tool was able to carry out a successful or unsuccessful evaluation compared to the total number of validation techniques applications. These metrics are aimed at making it clearer to the users that the tool is not able to decide on the accessibility of all the web elements analysed: thus, even if the Accessibility Percentage is 100%, users still have to check the Evaluation Completeness to understand to what extent the automatic validation has been able to decide on the correct application of all the validation checks. We have extended the tool to show the application of the metrics at both techniques and success criteria levels (see Fig.1). Each validation result is assigned a weight based on the corresponding WCAG conformance level of the applied validation technique: Level A: 1 (minimum level of accessibility), Level AA: 0.6, Level AAA: 0.2.

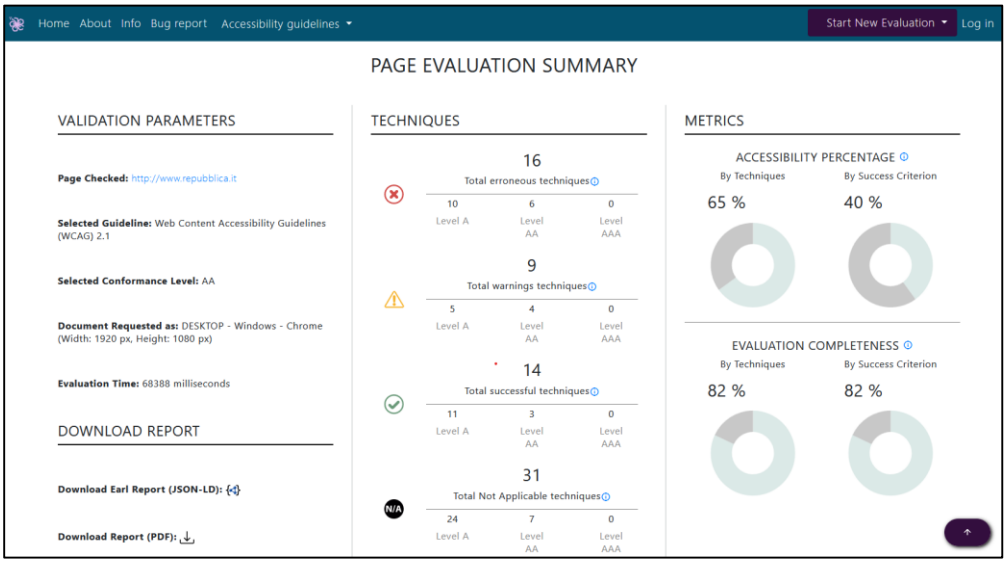

Figure 1: Page evaluation summary with the indication of metrics application at both techniques and success criteria levels.

3.2 Crawling Process and Dynamic Content Management

The first step in the validation process is to collect the web pages to validate. The goal of the crawling process is to discover the pages that should be validated starting from the home page (also known as the 'seed' page) of the considered website. For the crawling process, MAUVE++ exploited the Crwl4j library (https://github.com/yasserg/crawler4j), an open-source and multi-thread Java library which provides a simple interface for crawling the web. During the analysis of

the results of the crawling process, we realised that for several websites the crawler found only one page, namely the one provided as the seed URL. After analysing the target website by accessing it through the browser, we discovered that multiple links were available: thus we realized that the Crwl4j library does not implement Server Side Rendering (SSR) (i.e. rendering a client-side application directly on the web server), and consequently, it is not able to identify links that appear in the DOM of the page when it is loaded, but which are not included in the static HTML code. Indeed, such websites are developed through modern frameworks such as React.js, Vue.js, Angular or similar technologies and, as such, they exploit the Client Side Rendering (CSR) paradigm. With this paradigm, a web page has an almost empty HTML skeleton with some CSS and JavaScript: when the browser loads the page, the JavaScript code is executed to dynamically render the rest of the page by populating it client-side through REST services providing the information needed which are rendered within the initial HTML skeleton.

Thus, also for the crawling phase we had to find a library implementing the Server Side Rendering (SSR) paradigm, so as to find all the links within the considered pages. Actually, only for the validation process, MAUVE++ already considered a server-side rendering library in the form of an external Puppeteer service which receives the URL and some device target parameters (i.e. screen width and height) and returns back the HTML source code of the considered page. However, the first test for the large-scale validation highlighted a high latency due to the overhead in calling the external service. For this reason, we decided to integrate the Selenium Library (https://www.selenium.dev/) in the new version of MAUVE++, to support the server-side rendering both for crawling and validation processes.

Even though the Selenium library has been primarily developed for automating web applications for testing purposes, it is not limited to that. Indeed, to obtain a DOM wholly rendered on the server side, we exploited the Selenium WebDriver, a collection of language-specific bindings to drive a browser natively. The Selenium WebDriver supports several browsers (i.e. Chrome, Edge, Firefox, Safari) and supports programmatically accessing the server-side rendered DOM. Consequently, we moved from Crawl4j to the Selenium library, and we re-engineered the crawling process by implementing it as described in the following.

First of all, the seed page is loaded in a headless version of Chrome through the Selenium WebDriver. Then, it collects all the links belonging to the web page by adding the elements with tag ‘a’ and excluding all the links that do not belong to the same domain of the seed page and matching the extension to the list of extensions that do not represent an HTML page (e.g. jpg, svg, pdf, etc). If it does not find a sufficient number of links (a requirement of the large scale validation process was to evaluate at least 200 pages for each considered website), it adds the URLs discovered in the seed page to a First In First Out (FIFO) queue. This queue collects the list of the pages that should be visited until the crawler discovers a sufficient number of URLs. It is worth noting that not all the pages are loaded by the crawler since it usually finds the target number of URLs before visiting all of them. By loading all the discovered URLs we can assure that the page actually exists, however in this case the crawling process may require a significant amount of resources in terms of memory and time. For this reason, we decided to optimize the process by adding each link to the discovered page list without actually loading the associated page; this optimisation may introduce some "Page not found" errors during the validation process because among the collected links there could be some “broken links”; however, we calculated that their number is very low and the cost (in terms of time and memory resources) to visit each page is much higher, so we preferred running the risk of having some "Page not found" errors.

Even though the large-scale validation goal was to evaluate at least 200 pages for each considered website, we decided to collect 50 additional pages (thus, the crawler was configured to collect up to 250 pages) for two main reasons. The first one is that the crawling and validation phases are not immediately sequential, indeed the validation process may be performed even weeks after the web site has been crawled, thus among the discovered pages there can be some of them

that no longer exist. Secondly, during the validation process there can be issues related to the network, unreachable websites, target servers may be down or the HTML/CSS parser may throw an exception. Because of these reasons, we considered that 50 additional pages might be a sufficient number to manage run time issues.

The other crawler parameters set in the configuration are: *Politeness Delay* = 300ms; this parameter sets the interval between the requests, and we wanted to avoid frequent accesses to the considered websites, since they could raise some security exception; *Depth*: there is no limit in terms of the depth in the web site directory during the crawling; *Crawling Policy*: we decided to exclude links pointing to domains different from the seed URL; *Content-Type* = text/html: we have to evaluate webpages, thus all the URLs having a content-type different from text/html are ignored.

The result of the crawling phase is a list of URLs, which are passed to the validation part of the tool. Our server has two CPUs composed of 24 cores each, which support high parallelism for both the crawling and the validation activities. By taking into account the architectural settings of the CPUs installed in our server for the large-scale validation, we tuned the crawler parameters in order to maximise the parallelism and then obtain a high-performance crawling process. With 48 cores we are able to crawl in parallel 48 websites (48 thread controllers) and for each controller, the system executes the algorithm described above.

3.3 Validation Process

At the beginning of its execution, the validator retrieves from the database all the websites for which the crawler found at least 50 pages. The validator process is composed of different steps; 1) download the source code of the webpage; 2) provide the downloaded HTML to a parser which builds the webpage DOM; 3) analyse the DOM element to check whether they meet the WCAG success criteria.

The MAUVE++ validation is applied to one page at a time, thus, it has to call the validation engine multiple times in order to perform the validation of the 200 pages considered for each website. Similarly to what we did for the crawler, we set a politeness delay, by adding a time interval between the validation of different pages to avoid sending too many requests to the target server and making it unavailable. For several websites, we noticed that there were several *connection timeout* exceptions which actually blocked the validation execution since it was not possible to download the source code of the webpage to validate. We discovered that from the validator host it was not possible to access the whole website, while from different IPs address, there were no problems. Thus, we realized that some providers managing the hosting of several public administration websites inserted the IP of our server in a blacklist since they interpreted the crawler/validation activity as a "malicious request" or as a bot performing an attack from our host. In this case, we found a temporary solution by directing the network traffic toward a proxy so that we access the websites with an IP address different from the one blocked.

The most time and memory-consuming steps in large-scale validation involved:

1. the creation of Selenium WebDriver (which implies opening a headless version of Chrome browser),
2. loading the web page and getting the DOM, and finally,
3. closing the WebDriver.

Considering a multi-thread architecture of the server performing the whole process, steps 1 and 3 can be optimised by creating a pool of WebDrivers large as the number of threads that can be executed in parallel; so that when the validator finishes validating a website, the pool can be reused for the next web sites. The website validations were performed by running 8 controller threads (called *Evaluation Controller*) where each of them is in charge of managing the validation of a single website (Fig.2). Each *Evaluation Controller* receives the list of URLs discovered by the crawler and creates six *Evaluation Page Threads* in charge of evaluating one page each. This results in a balanced allocation of them to the 48

cores available. Since the validation of websites is independent, we do not need a synchronization mechanism between *Evaluation Controller* threads. On the contrary, each *Evaluation Controller* thread receives the validation report from the corresponding Evaluation Page Threads and saves it in the database; finally, it has to wait for the completion of all the validation threads to save the website validation summary on the database.

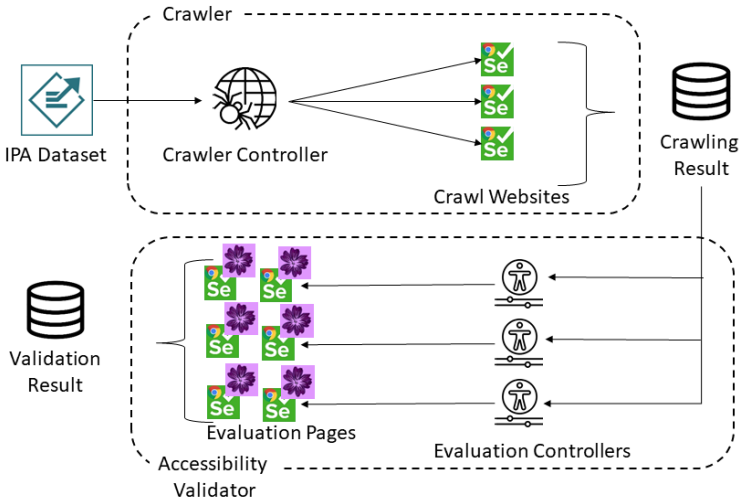

Figure 2: The crawling and validation process.

4 LARGE-SCALE VALIDATION RESULTS FOR ITALIAN PUBLIC ADMINISTRATION WEB SITES

The source for the reported Large Scale Validation is the public database called IPA (Index of Digital Domains of Public Administration - https://indicepa.gov.it/ipa-dati/dataset/enti) containing 22898 entries on January 30th, when we dumped it by downloading the associated CSV file. The IPA database contains 16 different fields ranging from a unique code (IPA code), website URL, mail address, ZIP code and so on. We extracted only 3 of such fields: the IPA code to uniquely identify the website, the URL and the ZIP code used to infer the corresponding region to perform georeferenced statistics later on. Among the 22898 entries available in the IPA database, we imported into our database 21932 websites, since 966 of such entries did not contain any URL or had a wrong format. The validation was performed in March 2023.

Table 1 presents the results obtained from the accessibility analysis. For each success criterion addressed by the tool (31 out of 50, only considering conformance levels A and AA), it shows the occurrences of Failure, Success, and cases where the tool was unable to provide a definite response (Cannot Tell).

By analysing the obtained results, it is possible to see some quantitative differences, even significant ones, between the occurrences of the various success criteria. These differences are also due to the different subsets of HTML tags for which the analyses of a success criterion are conducted. For example, success criterion 2.4.7 (“Focus Visible”), for which MAUVE++ implements the checks of techniques G195 and F78, has many occurrences both as successes and as failures

because the set of tags analysed includes <a>, <input>, <button>, which are generally more present compared to other HTML tags on a web page.

Table 1: The 20 success criteria with higher number of violations detected by MAUVE++

Success Criterion	Failures	Success	Cannot Tell
2.4.7 Focus Visible	163.031.627	179.224.514	0
1.4.1 Use of Color	96.191.514	109.400.350	0
1.4.3 Contrast (Minimum)	56.792.230	468.413.001	0
1.1.1 Non-text Content	46.675.335	29.420.357	18.223
1.3.1 Info and Relationships	28.215.948	29.969.030	19.705.932
2.4.4 Link Purpose (In Context)	24.296.529	311.470.139	814.261
4.1.1 Parsing	22.207.640	197.085.899	0
4.1.2 Name, Role, Value	21.264.521	70.131.938	58.692.006
1.4.11 Non-text Contrast	8.792.448	25.257.703	0
1.3.5 Identify Input Purpose	4.727.433	1.595.534	0
1.4.5 Images of Text	3.394.246	1.787.789	0
3.3.2 Labels or Instructions	1.972.564	10.521.129	9.637
1.4.10 Reflow	1.624.428	5.394.862	4.849
3.2.2 On Input	1.497.010	2.556.529	0
3.1.1 Language of Page	291.191	2.486.276	0
1.2.3 Audio Description or Media Alternative	52.974	6.242	0
1.4.12 Text Spacing	12.427	332	0
2.4.2 Page Titled	8.988	2.702.032	36
3.1.2 Language of Parts	4.957	2.511.775	0
2.2.1 Timing Adjustable	996	3.779	0

To highlight another opposite case, we can explain the low number of occurrences for criterion 2.2.1 by stating that the analysis of technique F41, implemented by MAUVE++ for this success criterion, only considers <meta> tags with the attribute "http-equiv" and the value "refresh" as reference tags. It is understandable that this tag may appear once per page or be completely absent.

The success criterion with the highest number of failure occurrences is criterion 2.4.7 Focus Visible, which belongs to the WCAG Operable principle . Keyboard navigation is a mode of navigating web pages where it is important to indicate which HTML element is selected. Indeed, unlike mouse navigation, it is not a pointer that defines the portion affected by a potential event, such as a click, but a clearly visible border around the concerned element. If this border is not present or barely visible, the use of the web resource for users who rely on an alternative navigation mode, such as keyboard navigation, becomes difficult, if not entirely compromised.

The next two success criteria with the highest number of error occurrences belong to the Perceivable principle. Specifically, criteria 1.4.1 and 1.4.3 aim to provide accessibility solutions to address certain issues related to visual perception. Regarding criterion 1.4.1 Use of Color (Level A), MAUVE++ implements the failure technique F73, associated

with creating links that are not visually evident without colour vision. The check is performed on all <a> HTML tags, for which certain CSS properties are analysed to verify if the interactive textual element is distinct from plain text. Criterion 1.4.3 Contrast (Minimum) (Level AA), although belonging to the Perceivable principle as criterion 1.4.1, suggests ensuring sufficient contrast between non-interactive text and the background on which this text is presented.

Table 2 shows the percentage values of the metrics calculated by MAUVE++, percentage of accessibility and percentage of completeness. For each region, it reports the average of the metrics applied to the Web sites considered. These results are aggregated for the region of Italy to which the analyzed websites belong.

Table 2: The metrics results calculated by MAUVE++ grouped by region

Region	Accessibility	Completness	Number of sites	Population
Abruzzo	76,75	80,24	457	1.269.860
Basilicata	75,32	80,51	228	536.659
Calabria	76,78	80,86	612	1.841.300
Campania	76,13	81,58	1031	5.592.175
Emilia-Romagna	77,35	80,35	852	4.426.929
Friuli-Venezia Giulia	74,17	77,61	405	1.192.191
Lazio	76,42	80,28	926	5.707.112
Liguria	75,26	80,91	390	1.502.624
Lombardia	76,28	80,79	2403	9.950.742
Marche	75,80	80,80	422	1.480.839
Molise	74,18	80,63	172	289.840
Piemonte	74,98	80,82	1461	4.240.736
Puglia	76,01	80,97	695	3.900.852
Sardegna	79,22	80,55	617	1.575.028
Sicilia	76,82	79,80	1042	4.802.016
Toscana	76,91	81,30	706	3.651.152
Trentino-Alto Adige	80,88	82,65	640	1.075.317
Umbria	75,87	80,27	196	854.137
Valle d'Aosta	71,92	81,71	129	122.955
Veneto	76,82	79,24	1103	4.838.253

5 CONCLUSIONS AND FUTURE WORK

In this study, we discuss how a tool for automatic Web accessibility validation (MAUVE++) has been extended to address large-scale accessibility validations, and the results that it provided when applied to the Web sites of Italian public administrations, thus analysing 2.7 million web pages to check their compliance to WCAG 2.1., thus providing the largest and most updated view of the accessibility situation of public websites in Italy, even if the tool was not able to validate all the sites for several reasons (connection problems, security issues, problems in the parsing of the content, …).

Such results have been used by the National Authority for accessibility (AGID) to provide information on the state of accessibility at the national level (https://accessibilita.agid.gov.it/). We indicate the most violated success criteria and how they have been assessed. Such data are important to understand the actual accessibility levels of the Web applications made available to the public, and identify areas that need to be better addressed by Web developers and designers.

Future work will be dedicated to including in the analysis also the PDF files linked to the Web pages, providing support also for the WCAG 2.2 version and the ACT Rules, developing a plugin for the WordPress CMS to support validation in the Web development phase, and integrating results of manual evaluations for the cases in which the tool is not able to provide definitive results.

ACKNOWLEDGMENTS

This work has been partially supported by the Italian PNRR 1.4.2. We thank AGID for the useful discussions.

REFERENCES

Julio Abascal, Myriam Arrue & Xabier Valencia (2019). Tools for Web accessibility evaluation. Web Accessibility (pp. 479-503). London: Springer.

Siddikjon G. Abduganiev. 2017. Towards automated Web accessibility evaluation: a comparative study. Int. J. Inf. Technol. Comput. Sci.(IJITCS) 9, 9 (2017), 18–44.

Abdo Beirekdar, Marc Keita, Monique Noirhomme, Frédéric Randolet, Jean Vanderdonckt & Céline Mariage. 2005. Flexible reporting for automated usability and accessibility evaluation of web sites. In Human-Computer Interaction - INTERACT 2005. Lecture Notes in Computer Science, Costabile M. F. and Paternò F.(Eds.). Springer, Berlin, 3585.

Giovanna Broccia, Marco Manca, Fabio Paternò, Francesca Pulina. Flexible Automatic Support for Web Accessibility Validation. Proc. ACM Hum. Comput. Interact. 4(EICS): 83:1-83:24 (2020)

Andreas Burkard, Gottfried Zimmermann and Bettina Schwarzer. 2021. Monitoring Systems for Checking Websites on Accessibility. Frontiers in Computer Science 3 (2021), 2.

EU Commission. (2016, October 26). Directive (EU) 2016/2102 of the European Parliament and of the Council. Retrieved from https://eur-lex.europa.eu: https://eur-lex.europa.eu/eli/dir/2016/2102/oj

Nicola Iannuzzi, Marco Manca, Fabio Paternò, Carmen Santoro, Usability and transparency in the design of a tool for automatic support for web accessibility validation, Universal Access in the Information Society, 2022, 1-20, Springer Verlag.

Yavuz Inal, Deepti Mishra, and Anne Britt Torkildsby. 2022. An Analysis of Web Content Accessibility of Municipality Websites for People with Disabilities in Norway: Web Accessibility of Norwegian Municipality Websites. In Nordic Human-Computer Interaction Conference (NordiCHI '22). Association for Computing Machinery, New York, NY, USA, Article 65, 1–12. https://doi.org/10.1145/3546155.3547272

Marco Manca, Vanessa Palumbo, Fabio Paternò, Carmen Santoro, The Transparency of Automatic Web Accessibility Evaluation Tools: Design Criteria, State of the Art, and User Perception, ACM Transaction on Accessible Computing, ACM Press, 2022

Beatriz Martins, Carlos Duarte. Large-scale study of web accessibility metrics. Univ Access Inf Soc (2022). https://doi.org/10.1007/s10209-022-00956-x

Ombretta Gaggi and Lorenzo Perinello. 2022. Improving accessibility of web accessibility rules. In Proceedings of the 2022 ACM Conference on Information Technology for Social Good (GoodIT '22). Association for Computing Machinery, New York, NY, USA, 167–174. https://doi.org/10.1145/3524458.3547267

Silvia Mirri, Ludovico A. Muratori, & Paola Salomoni. (2011). Monitoring accessibility: large scale evaluations at a geo political level. The proceedings of the 13th international ACM SIGACCESS conference on Computers and accessibility (pp. 163-170). New York: ACM.

Marian Pădure, and Costin Pribeanu. 2019. Exploring the differences between five accessibility evaluation tools. (2019).

Fabio Paternò, Francesca Pulina, Carmen Santoro, Henrike Gappa, Yehya Mohamad. (2020) Requirements for Large Scale Web Accessibility Evaluation. In: Miesenberger K., Manduchi R., Covarrubias Rodriguez M., Peňáz P. (eds) Computers Helping People with Special Needs. ICCHP 2020. Lecture Notes in Computer Science, vol 12376. Springer, Cham. https://doi.org/10.1007/978-3-030-58796-3_33

Jens Pelzetter, A Declarative Model for Web Accessibility Requirements and its Implementation. Frontiers Comput. Sci. 3: 605772 (2021)

Christopher Power, André Freire, Helen Petrie, David Swallow. (2012). Guidelines are only half of the story: accessibility problems encountered by blind users on the Web. Proceedings of the SIGCHI conference on human factors in computing systems (pp. 433-442). ACM.

Costin Pribeanu. (2019). Large-scale accessibility evaluation of Romanian municipal websites. Int. Journal of User-System Interaction, 12(2), pp. 83-98.

Antonio G. Schiavone, Fabio Paternò, An extensible environment for guideline-based accessibility evaluation of dynamic Web applications, Universal Access in the Information Society, Springer Verlag, 14(1): 111-132, 2015.

Markel Vigo, Justin Brown, Vivienne Conway. (2013). "Benchmarking web accessibility evaluation tools: measuring the harm of sole reliance on automated tests", in Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility, 1–10.

Yeliz Yesilada, Giorgio Brajnik, Markel Vigo, Simon Harper: Exploring perceptions of Web accessibility: a survey approach. Behav. Inf. Technol. 34(2), 119–134 (2015)