LAB University of Applied Sciences

ACCESSIBILITY TESTING OPPORTUNITIES IN WEB DEVELOPMENT

LAB University of Applied Sciences
Bachelor of Business Administration
Degree programme in Business Information Technology
Autumn 2020
Alexis Oksi

Abstract

Author	Type of publication	Published
Oksi, Alexis	Bachelor's thesis	Autumn 2020
	Number of pages	
	38	

Title of publication

Accessibility Testing Opportunities in Web Development

Name of Degree

Bachelor of Business Administration

Abstract

Web accessibility has quickly become an important topic among creators and maintainers of websites, since new laws on web accessibility have been passed around Europe imposing accessibility requirements on many websites as well as spreading awareness about the topic in the industry rapidly. At the same time the question among development teams arises on how to best ensure that these newly introduced requirements are met. By exposing relationships between existing testing practices in web development and available accessibility testing practices, this study has aimed to identify opportunities to integrate accessibility testing among other testing practices in the web development industry.

This study has applied a literature review as well as qualitative research methods. The literature review was chosen to expose existing knowledge and to learn about the topic. A semi-structured interview approach was used as a data collection tool in order to enable a dialog between interviewer and interviewee and bring up information that would have otherwise remained uncovered. The data was gathered from two developers that had experience with web accessibility and public sector clients, who are affected the most by the new legislation. The data was coded to be able to perform a meaningful analysis and recognize patterns and relationships.

As a result, this study was able to identify several software testing practices that have been applied in web development. Similarly, web accessibility testing practices were classified as software testing practices and an overlap between these and existing testing practices was found. Further research is required to evaluate if integrating accessibility testing methods among other similar practices would prove to be the most efficient and effective approach.

Keywords

accessibility, web accessibility, web development, testing, WCAG, software testing

CONTENTS

1 INTRODUCTION

> *People with disabilities can and do surf the Web, often with the use of adaptive technology that compensates for particular disabilities. But for Web sites to be reasonably accessible, Web authors have to take certain care in the way they create pages. (Clark 2003)*

As Clark (2003) puts it aptly, web accessibility is about taking into consideration disabled users which contrary to popular belief are oftentimes quite capable when acting in the Web. To not create any barriers for these users and their assistive technologies, the owners, designers, developers and authors of the web have to take all users, including disabled ones, into consideration.

Having a background as web developer I have had the pleasure of working with a long-term client which has had the requirement to fulfill web accessibility standards for multiple years already, which is not very common in Finland. During this experience I have learned about accessibility in general, web accessibility specifically and about lots of varying situations where accessibility needs to be considered in a digital service. In addition, sharing knowledge about web accessibility and how to ensure meeting these standards continuously have become an important part of my work as a developer, consultant and team lead.

I have followed the passing of new directives and laws concerning web accessibility in the EU with great interest, as the effort of meeting accessibility standards will, also in Finland, become relevant for a large number of organizations and their digital services. In that context, the question of how to ensure that these standards are met will be raised. In my experience the question of what is accessible or inaccessible, or if something is accessible enough, is very difficult and can sometimes mean compromises of some sort. I have encountered development teams that use methodologies and processes where testing is already an integral part, other teams or individuals perform testing more loosely. For many organizations wanting to meet accessibility standards the question of how to test and determine the level of accessibility will become important, which is why I chose this topic.

This paper aims to identify opportunities where accessibility testing can be integrated in an existing or new process of developing and especially testing a website or web service. Considering the small scope of this thesis, the focus will mostly be on development and not, or only slightly, cover other important areas of creating and maintaining a web service, such as user research, visual design, content design and content authoring.

1.1 Research objectives, questions and limitations

When looking into existing literature on the areas of software testing, web accessibility testing and web accessibility in general one can see these areas have been studied to some extent, even though mostly in separate entities. The intersection of how web accessibility should be tested and how this testing could be integrated in an existing or new testing process does however not yield a lot of knowledge. While some sort of testing is almost always done, it can be unclear on when, how and where accessibility testing should take place. To identify web accessibility testing methodologies and their applicability I will aim to answer the following main research question:

- How could web accessibility testing methods be integrated within existing web development testing processes?

To be able to answer this question, the following sub-questions need to be addressed:

1. What web accessibility testing methods are available and how are they applied?
2. What testing methodologies are available in software development?
3. How is testing already being done in development teams that are affected by accessibility legislation?

The research will be limited to a sample of developers from teams that are or have been developing publicly funded websites, as those are affected by accessibility legislation and might differ from privately funded businesses. The focus will remain on the testing done by or for developers, as including the areas of design, copywriting and maintenance would not fit within the scope of this thesis. All areas might however be covered partly, whenever sensible. As laws and monitoring, as well as the culture of developing and testing is likely to be quite different across Europe and the World, this research will be limited to Finland to be able to gather reproduceable results and yield valuable results to Finnish organizations.

In Figure 1, the different topic areas and their intersections are visualized. Currently applied testing methods, meaning the subset of testing methods a development team is currently using, are likely to be partly classifiable as software testing methods. Some web accessibility evaluation methods are also likely to be classifiable as software testing methods This research aims to identify the intersection of these three groups of testing methods, the currently applied testing methods, software testing methods more generally and web accessibility testing methods.

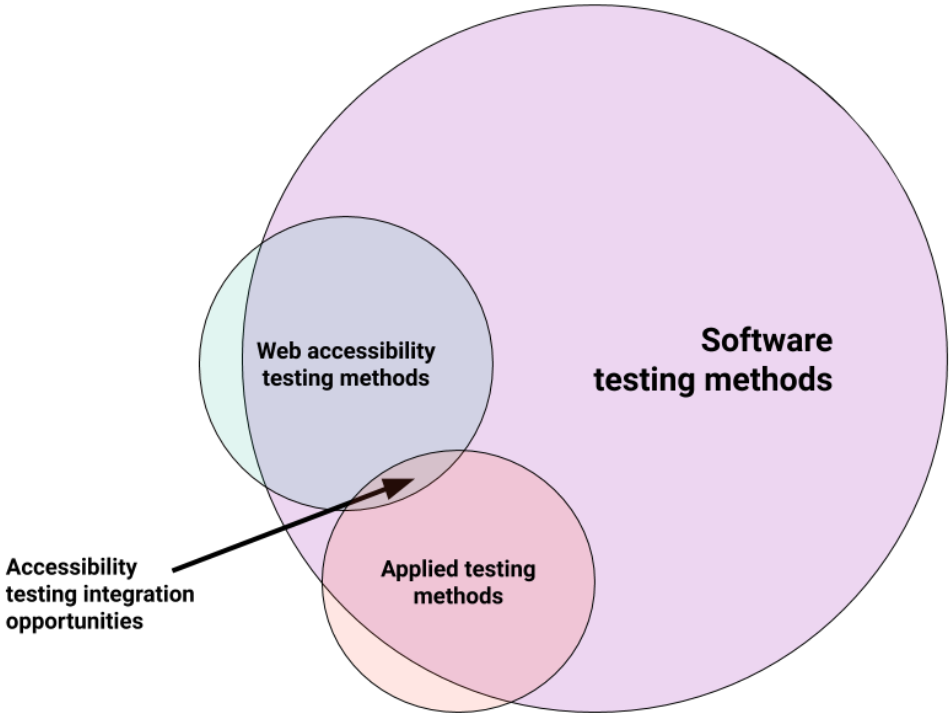


Figure 1. Venn diagram on intersection of testing methods.

To conclude, the objective of this research is to identify opportunities to apply web accessibility testing methods within readily used testing practices.

1.2 Research methodology and data collection

The first two sub-questions in this research, available accessibility testing methodologies as well as available software testing methodologies, have been addressed extensively in existing literature. To introduce these available methodologies and explain the purpose of using them, I chose to conduct a literature review in the first part of my research. As Rowland (2014) suggests, a review of previous literature can help the reader understand the reasoning behind a research and expose existing knowledge. In addition, a literature review will assist the author in learning and thinking critically about the topic at hand.

1.2.1 Semi-structured interviews

The third sub-questions, identify existing testing processes in teams that are affected by web accessibility legislation, has no pre-existing data or literature that could be used. As no secondary data is available, I chose to gather the necessary data through qualitative research methods, as the answer to the research question is not quantifiable. To gather this data, I chose to conduct semi-structured interviews with developers or testers from

affected teams, as Saaranen-Kauppinen and Puusniekka (2009, 52-53) suggest that this methodology can yield profound and diverse information around a topic.

The question of how testing is currently being done, does not state a clear hypothesis, but is set to look at the collected data without being certain what that data will conceal. According to Saaranen-Kauppinen and Puusniekka (2009, 6-7) the lack of a clear hypothesis and data-driven research are factors to categorize this research as qualitative. Typical methods for gathering information in qualitative research are observations and interviews, or at present the use of existing material and documents as well (Saaranen-Kauppinen & Puusniekka 2009, 47). An observative approach could yield deep insights in the existing testing practices in an organization, but would restrict the data collection to one or two development processes due to the small scope of this research. Due to the lack of existing material of the organizations in question I chose to conduct interviews instead, as I expect to gather a good understanding of existing testing processes in different types of relevant organizations.

Saaranen-Kauppinen and Puusniekka (2009, 52-53) introduce interviews as the most used data collection method in qualitative research and state that in ideal conditions a researcher can gather diverse and profound information through interviews. They categorize several types of interviews in the categories of structured interviews, consisting of strictly defined questions and readily provided options for answers, and semi-structured or informal interviews, where no answers are provided, and the presentation of the questions is more flexible. I chose to conduct semi-structured interviews in this research, as this method enables a dialog between the interviewer and the interviewee, which can bring up information that would have remained uncovered in a more structured approach (Saaranen-Kauppinen & Puusniekka 2009, 56). From my own experience the topic of testing can also lead to emotional reactions, as developers get the feeling they should be testing more or differently already. To avoid creating pressure and pre-judging what testing could or should be done a focused interview method is helpful, as open-ended questions can be used to create a natural dialog.

1.3 Structure

This thesis is divided in five sections, starting with an introduction on the research questions, objectives and restrictions as well as the research methodology and data collection. The second section includes the conceptual framework covering web accessibility, in an effort to expose information about web accessibility guidelines and evaluation methods, as well as the topic of software testing, including classifications and definitions of testing levels and methods. The third section covers the data collection,

including interview themes, participants and execution, and the data analysis around the topics of testing practices in web development and how they compare to software testing and accessibility testing. In the fourth section conclusions based on the data analysis are being drawn. Finally, the findings are discussed by looking at industry and research recommendations in the fifth and final section of this report.

2 CONCEPTUAL FRAMEWORK

2.1 Research topic definitions

Web Accessibility

The term accessibility can be defined as a quality of being able to reach or enter something, sometimes concentrating on the ease of use for people who have a disability (Online English Dictionary 2017). Web accessibility focuses on accessibility in the context of the web and means that people with disabilities can use the web. Using the web means perceiving, understanding, navigating and interacting with the web. (Henry 2005.)

Web Accessibility policies

Accessibility in the web is by no means a recently emerged field, but has been around since 1996 when the World Wide Web Consortium started a Web Accessibility project (Dardailler 2009). It has however gained popularity in recent years, as more laws and policies have been passed around the world. The list of laws has grown rapidly in the last two years, after a directive by the European parliament (from here on “EU directive”) was passed in 2016, which affects all countries in the European Union (EU). This directive brings about laws concerning web accessibility in all EU member countries. (Mueller et al. 2018.) In April 2019 the Finnish law concerning the EU directive (Laki digitaalisten palvelujen tarjoamisesta 306/2019) came into effect. This law will, among other things, require most public and some private sector websites to meet web accessibility standards (Aluehallintovirasto 2019). To ensure that the guidelines are met, website owners will have to design and develop according to these standards and report on their compliance visibly on the website. This means that all of these websites will benefit greatly from testing practices that help to meet the guidelines and enable reporting on compliance.

The recent rise in laws being passed is showing in a rising amount of research being done in the field of web accessibility. Mitronen (2018) for instance analyzed the effect the EU directive will have on websites. Mitronen lined out the status of the web accessibility legislation and which organizations will be affected by it and when trying to test an existing website, she noticed accessibility flaws which were not immediately noticeable to the regular user and found that a lot of content will remain inaccessible to disabled users. During her research she found that developing accessible websites isn’t necessarily expensive, but can become expensive when the accessibility requirements have to be applied at the very end of a development process. This confirms my assumption that the chosen topic is currently quite relevant and important.

Web Content Accessibility Guidelines (WCAG)

Initiatives such as the Web Accessibility Initiative (WAI), founded by the World Wide Web Consortium (W3C 2017), are advocating for a more accessible web by developing strategies, guidelines and resources for Web Accessibility. The WAI is contributing significantly by developing the Web Content Accessibility Guidelines (WCAG). The newest finished version of these guidelines is WCAG 2.1, which is also incorporated in the European standard EN 301 549. The EN standard references almost directly the WCAG 2.1 levels A and AA guidelines, which means these guidelines can effectively be used as a reference point to develop an accessible website when trying to conform to the EU directive (Aluehallintovirasto 2019). The WCAG provide detailed information on what steps need to be taken and what things should be considered when developing accessible websites. In addition to incorporating accessible design recommendations in the planning and development of a website, its accessibility needs to be evaluated throughout the lifecycle to determine if it is accessible. (Henry 2005.)

Apart from the EN standard referenced in the EU directive, affecting all countries in the EU, many other countries have legislation on web accessibility. For instance, Section 508 in the USA, or Regulations on Universal Design of ICT in Norway, all of which make use of the WCAG or a derivative of it (Mueller et al. 2018). Some older policies do reference the older version WCAG 2.0. The new version WCAG 2.1 includes however the exact same requirements as 2.0 and only introduces additional requirements. This means that content that conforms to WCAG 2.1 also conforms to WCAG 2.0, so WCAG 2.1 can be used as a standard even though a policy would reference the older 2.0 standard. (Henry 2018.)

The focus in this research will remain on evaluating and testing for WCAG conformance, as that will be the standard used to measure web accessibility in Finland. It is however important to note that WCAG might leave important barriers uncovered, especially if they are hard or impossible to measure or test.

Software testing

In the process of developing a website, similar to other areas of software development, testing already plays an important role. Depending on the scope and requirements of the development, a combination of automated and manual testing methods can be used to ensure that the released product performs its task as intended (Homès 2013). Homès (2013, 1) describes testing as necessary to avoid failures visible to customers. Software testing practices, which essentially ensure that the developed software meets requirements and have no failures, have been studied extensively and lots of literature on

the topic is available. In a new publication by Juvonen (2018, 25-36) the necessity of testing and different forms of software testing are described. Testing should be performed even before development is started, by creating a wireframe that demonstrates that requirements can or cannot be met. When looking into testing methodologies during development, the testing can be done from different standpoints, phases and scopes, to ensure that testing is done early on and often. Juvonen concludes that in general, a fitting testing methodology needs to be chosen depending on the project, as there are many varying factors such as budget, timetables, quality requirements, company culture and so on. In an effort to reduce testing costs, automated testing methods are essential and can free up time in areas of testing where no or little automation is currently available.

2.2 Web accessibility

When looking to make the web accessible for all users, including disabled users, a general understanding of common disabilities is necessary. There is a diversity of reasons why a person might have impaired abilities, some of which are age-related, health conditions, temporary and situational impairments. Forms of disabilities cover auditory, cognitive, physical, speech and visual impairments.

To illustrate the barriers disabled users might encounter Abou-Zahra (2017a) highlights some examples of disabilities and barriers. Examples of auditory disabilities are deafness, being hard of hearing or deaf-blindness, and these users have trouble consuming audio content or services that rely on using voice only. Cognitive disabilities include a large range of disabilities, for instance ADHD (Attention deficit hyperactivity disorder), learning disabilities and memory impairments only to name a few. This group of users experience barriers when encountering moving content, long and complicated text, complex navigations and interfaces that require the user to remember information. Physical or motor disabilities can for example mean an amputation, rheumatism or tremors. Users with physical disabilities encounter barriers when keyboard navigation and input are not completely supported or if there are unpredictable navigation mechanisms. An example of a speech disability is muteness and a barrier can be the requirement of speech input or services that offer phone calls as the only way to communicate. Visual disabilities can vary from color blindness or low vision to complete blindness. Barriers for this group of people can be either visual, such as no possibility to resize content and insufficient contrast, or technical, such as missing text alternatives that would be read out by assistive technologies to the user. (Abou-Zahra 2017a.)

Taking disabled users into consideration when designing and developing a service or website, will help substantially to minimize the number of barriers for these users. To

assist and standardize these efforts, the WAI is maintaining the WCAG, which can be helpful to when creating and auditing with accessibility in mind.

2.2.1 Understanding WCAG

As briefly mentioned in the introduction, the Web Content Accessibility Guidelines (WCAG) provide detailed information on what steps need to be taken and what things should be considered when developing accessible websites. (Henry 2005.) In its newest version 2.1 the guidelines consist of four principles. Under these principles are 13 guidelines, which are not testable, but are meant to help understanding success criteria and implementation techniques. For each of these guidelines there are testable success criteria, divided on three levels of conformance: A (lowest), AA and AAA (highest). Level AA is most often used as the required conformance level and consist of 50 success criteria. Each success criterion is also equipped with examples and techniques that can help to understand and meet the success criterion at hand. (Kirkpatrick et al. 2018.) Figure 2 illustrates the layers of principles, guidelines, success criteria and techniques of the WCAG 2.1 by the example of the level AA.

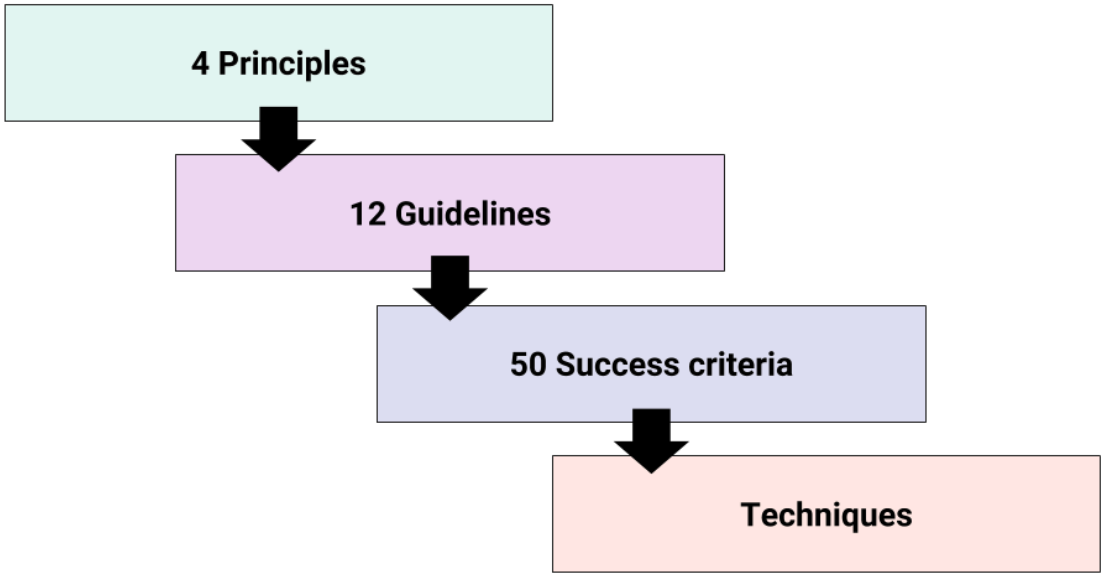


Figure 2. Waterfall diagram illustrating the layers of WCAG 2.1 Level AA.

Even though the WCAG standards have been referenced by most authors giving recommendations on accessible web design (Ng 2017, Riley-Huff 2012, Logacheva 2016), Friedman and Bryen (2007) criticize the lack of focus on cognitive disabilities and demand a higher priority of cognitive disabilities and suggest that further research is

needed to identify further barriers and show the benefits of removing those. The editors of the WCAG emphasize out themselves that even though the guidelines cover a large range of disabilities, they are not able to address all types, degrees and forms of disabilities (Kirkpatrick et al. 2018).

The four principles, containing the guidelines and success criteria, are *perceivable*, *operable*, *understandable* and *robust*. To give a basic understanding of the WCAG and areas it covers, I will give some examples for each of the principles.

Perceivable

Perceivability describes the goal to present information in a way that users can perceive it, ensuring that it's not invisible to all their senses (Cooper et al. 2017). Riley-Huff (2012, 33) mentions multiple practices to improve perceivability, for instance the use of big enough font sizes, at least 12 points or 14 points, as well as using few and readable fonts. Also, using uppercase makes text harder to read and high contrast is necessary to ensure readability. Using colors to convey meaning can be problematic for colorblind users and should not be used as only delimiter. (Riley-Huff 2012, 33.) Another important example of a perceivable success criterion is writing the markup for the content (HTML being the language used for that) using semantic HTML elements, especially for headings, as this makes understanding the content faster for all users, especially the ones using assistive technologies (Ng 2017).

For audiovisual content the content needs to be offered in an alternative manner ensuring that users with restricted sensory abilities can access the content in question. According to Ng (2017), using alternative text attributes for images is vital, as well as not having text on images. Charts and datasets should be summarized or presented in another way is possible. Purely decorative images do not have to have alternative texts, however. For images and videos text transcripts or even closed or open captions should be offered. These are also useful for accessing that content in a noisy environment or speakers of a foreign language. Embedded media from third parties should have a link to their source if possible, as the proprietary players often have more accessibility features available. Finally, disabling auto play is vital to give the user more power on how to interact with the content depending on individual needs. (Ng 2017.)

Operable

The principle of operability aims to ensure that users can operate the interface and the interface does not require interaction that a user cannot perform (Cooper et al. 2017). Logacheva (2016, 20) suggests designing big enough click targets and enough spacing

between different targets to make clicking on the right elements easier. Operating the interface should also not be restricted to keyboard or mouse users, but should work with either (Caldwell et al. 2008).

Understandable

Understandability means that users must be able to understand the information as well as the operation of the user interface (Cooper et al. 2017). As Riley-Huff put it: “Writing the Web is an art, and the style is minimalism” (Riley-Huff 2012, 31). When creating content for the web understandability should be of high concern. To achieve that idioms, double meanings and nuanced language should be avoided. It is recommended however to be succinct, to chunk content, use headers and write clear language. (Riley-Huff 2012, 31.) Ng (2017) also suggests using clear textual descriptions for links, instead “Read more” or “Info” texts that make understanding them slower and harder. She also recommends using short paragraphs, simple language with everyday words and formatting text to include considerable white space. (Ng 2017.) Also, the operation of the user interface should stay easy to understand, which can be achieved by dividing complex tasks into smaller ones and minimize the occurrence and consequences of errors (Logacheva 2016, 9-11).

Robust

Robustness means that content can be interpreted reliably by a variety of user agents, meaning the technologies that users use to access content (Cooper et al. 2017). This focuses mostly on the technologies being used and ensuring that they are use according to specifications and keeping different user agents in mind. Riley-Huff (2012, 31) mentions using semantic HTML and CSS, as well as making sure that JavaScript is used mostly as an enhancement and accessing the content is not dependent on it.

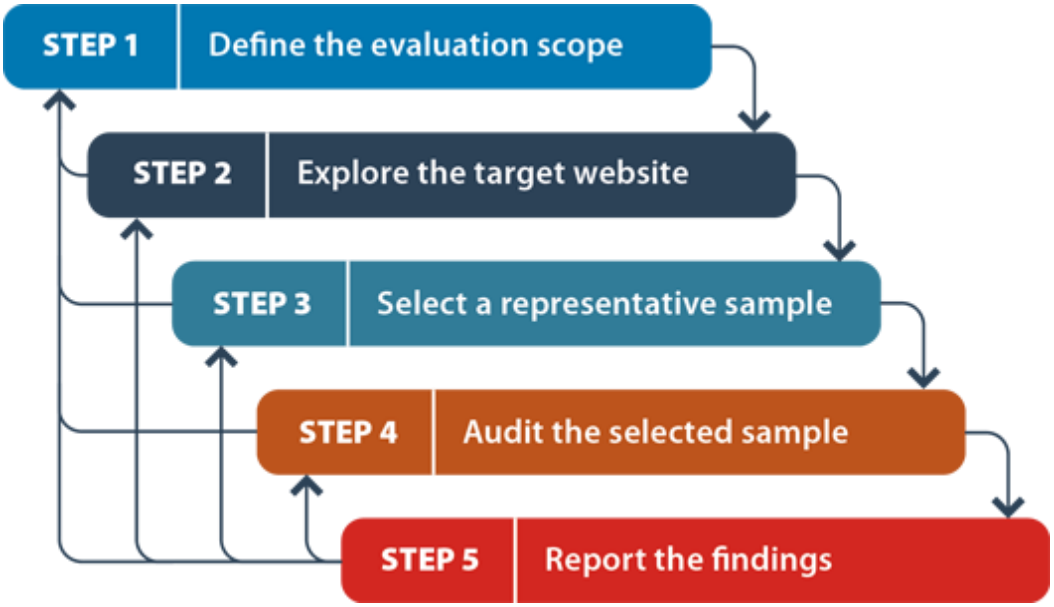
2.3 Accessibility evaluation

The WCAG guidelines were specifically designed as testable criteria, so that anyone can determine if something is accessible or not. This can be helpful for designers, developers and authors to check if their website will be accessible, but it can also be a tool for the product owners to determine if the deliverable meets requirements that were set. As some website are required to be accessible by law, the WCAG can help the monitoring body make the determination if that requirement is met.

2.3.1 WCAG-EM

In addition to the WCAG the WAI has also created a method to evaluate how well a website conforms to the WCAG, namely the Website Accessibility Conformance Evaluation Methodology (WCAG-EM). It can be applied to all websites and is meant as a methodology for thorough review of a website, but does require some expertise by its user in the areas of WCAG, accessible web design, assistive technologies and how people with different disabilities use the Web. (Henry & Abou-Zahra 2016). WCAG-EM is not meant to replace quality assurance measures that are necessary throughout design, development and maintenance processes and does not directly result in WCAG conformance if used on its own. Instructions for evaluating a website feature by feature are already covered by the WCAG, but the WCAG-EM describes a procedure to evaluate websites and establishes standardized best practices. (Velleman & Abou-Zahra 2014).

The procedure to evaluate conformance is divided into five main sections. First the scope of the evaluation is defined, so what is included, what the goal is and what conformance level is being used. Next the exploration of the website identifies key pages, functionalities and types of content. Based on this exploration representative samples are selected, both randomly and structurally, assuming not every web page can be included in the evaluation. The fourth step is the actual evaluation by checking for failures in meeting WCAG and measuring the accessibility support. As the last step a report is aggregated to communicate the findings. (Henry & Abou-Zahra 2016.)



Picture 1. Diagram about the iterations between the steps in WCAG-EM. (Velleman & Abou-Zahra 2014). Copyright © 2014 W3C® (MIT, ERCIM, Keio, Beihang), All Rights Reserved. W3C liability, trademark and document use rules apply.

To conclude, the WCAG-EM offers a procedure to select a representative sample of a website and report WCAG conformance. It does however require expertise and using it for evaluation does not directly result in conformance. The WCAM-EM could prove useful when starting to make improvements to an existing website or determining the level of compliance, as it will reveal out the most common and important WCAG failures.

Evaluation tools

To assist in an accessibility evaluation a plethora of different tools can be found and used. These tools check for different guidelines, WCAG 2.0 being the most popular one, and can work as a browser plugin, online tool, desktop application, API (Application Programming Interface), command line tool, authoring tool plugin or a mobile application. The licenses of these tools range from free, open source to enterprise and commercial ones. All tools are restricted to be of *assistance* in an accessibility evaluation and cannot replace human judgement, which is necessary for the actual determination. The tools can currently not cover all WCAG rules and they can often produce misleading or even false results. Tools can however be helpful in a manual review and reduce the time spent on an evaluation. (Abou-Zahra, 2017b.)

Abou-Zahra (2017b) recommends that each organization, project and team should try to find a tool that fits their needs, be it a tool assessing the code for developers or a plugin that helps content authors identifying accessibility checks.

In addition to existing tools, it should be noted that the European Commissioned co-founded the program Horizon 2020, which includes the WAI-Tools project. This project aims to develop, deploy and integrate test rules, which will further improve and standardize automated testing capabilities and most likely yield in more capable tools being available to both the monitoring bodies and the product owners creating accessible websites. (W3C, 2019.)

2.3.2 Expert testing

As the previously mentioned automated evaluation techniques are only meant as an assistance to human judgement, the importance of that human judgement becomes clear. To be able to make this judgement Brewer (2002) makes some recommendations on the necessary expertise. Brewer mentions expertise in web technologies, validation tools for web technologies, WCAG and related techniques, web accessibility evaluation approaches and usage, disability barriers, assistive technologies and adaptive strategies as well as involvement of people with disabilities in the evaluation. The expertise in these areas does not have to be concentrated into one individual in a project or organization, but can also be covered in a collaborative approach. A collaborative approach could for instance combine in-house experts with outside experts where needed. Brewer also names examples for this collaborative process, such as web developers from different units in a large corporation, or a small business who provides accessibility evaluation services with a multi-disciplinary team.

Brajnik, Yesilada and Harper (2011) studied if expertise in accessibility evaluation matters and found that depending on the metric used it does. They also found that single expert can identify about 72% of accessibility problems, two can reach 94% and three experts can cover all of the problems. In comparison at least 14 nonexperts would be necessary to identify all true problems and one expert could only identify about an average of 50% of the problems. This study also shows that experts spend much less time on the evaluation while being more confident on the results.

Expert testing can therefore mean that an evaluation is performed by a single expert or a more collaborative approach where experts from different disciplines come together, possibly also including disabled users.

2.3.3 EU directive monitoring

Apart from ruling that certain websites have to be accessible, the EU directive also requires regular monitoring and reporting of website accessibility by the Member States. These reports will be public and delivered to the European Commission. (European Commission 2018). This means that every member state will have to monitor and measure WCAG conformance on a regular basis and website owners will have to be prepared to be audited.

In an effort to find a standardized way of monitoring and reporting, the EU has commissioned an extensive study on monitoring methodologies for web accessibility. The main objectives of that study were providing a description of the current state of web accessibility monitoring, validating a set of monitoring methodologies to find out how reliable these methods are and giving practical recommendations on choosing web accessibility monitoring methodologies. These recommendations were based on the first two objectives in that study. As an outcome of the study, the authors did not find a suitable European-wide monitoring scheme, but found the current methods being used in Norway the most comprehensive, thus taking the Norwegian method as main reference for a European wide monitoring methodology. (Laurin et al. 2016.)

Based on this study the EU drafted and passed an implementing decision for the accessibility EU directive (Commission Implementing Decision (EU) 2018/1524). This decision will require member states to have two types of monitoring, in-depth monitoring and simplified monitoring. In an effort to not impose barriers on the market, the implementing decision does not require a specific test, assessment tool, browser, operating system or assistive technology. The focus of this decision lies on the monitoring methodologies and reporting instructions.

In-depth monitoring

The in-depth monitoring is done to a smaller sample of websites (exact number depends on the number of inhabitants of each member state). In-depth monitoring means that a thorough evaluation of a website is performed, which should include all steps of a process and should evaluate at least the interactions with forms, interface controls, dialog boxes and also confirmations for data entry and error messages. In addition, it can include usability tests, where a disabled user evaluates how complex it is to user the website. (Commission Implementing Decision (EU) 2018/1524.)

Simplified monitoring

Simplified monitoring is done to a larger sample and covers only a subset of the requirements in the standard. This simplified monitoring method is meant to be done using automated tests and should cover user accessibility needs to the maximum extent that is possible using automated tests, even though non-automated tests are allowed as well. The implementing decision defines the following disabilities which needs are meant to be covered:

- No or limited vision
- No perception of color
- No or limited hearing
- No vocal capability
- Limited manipulation or strength
- Photosensitive seizure triggers
- Limited cognition

(Commission Implementing Decision (EU) 2018/1524.)

To conclude, the EU directive and the related implementing instructions require the member states to perform two types of monitoring, automated and expert evaluation. As no tools or technologies are defined, each member state can decide the exact tools and techniques themselves and can also renew them freely. This freedom will enable the monitoring body to apply and update the exact monitoring details depending on the changing needs of the disabled population and services made available to them. That same freedom will however also mean unclear monitoring for the owners, designers, developers and maintainers of web services, which means a great responsibility in incorporating accessibility requirements throughout the development cycle.

2.4 Software Testing

Software development can be complex and prone to errors and faults might appear at any point during the design, development and maintenance of a software. In the process of this development, efforts are set to meet desired requirements the software should fulfill. These efforts can be categorized to efforts contributing to the construction of the software and efforts aiming at checking the quality of the development process and the produced software. All of these efforts related to quality are also referred to as quality process or

quality assurance. This process spans over the entire development cycle and testing efforts are ideally started early on and performed often. Activities within the quality process aim to either reveal faults or assess the readiness of the software. (Baresi & Pezzè 2006, 89-91.)

2.4.1 Testing levels

Baresi and Pezzè (2006) describe how in software development the entire system can be split into smaller abstraction levels, namely modules or components, subsystems and the final system. Similarly, the testing can be performed at these different levels and is categorized into module or unit, integration, system and acceptance testing.

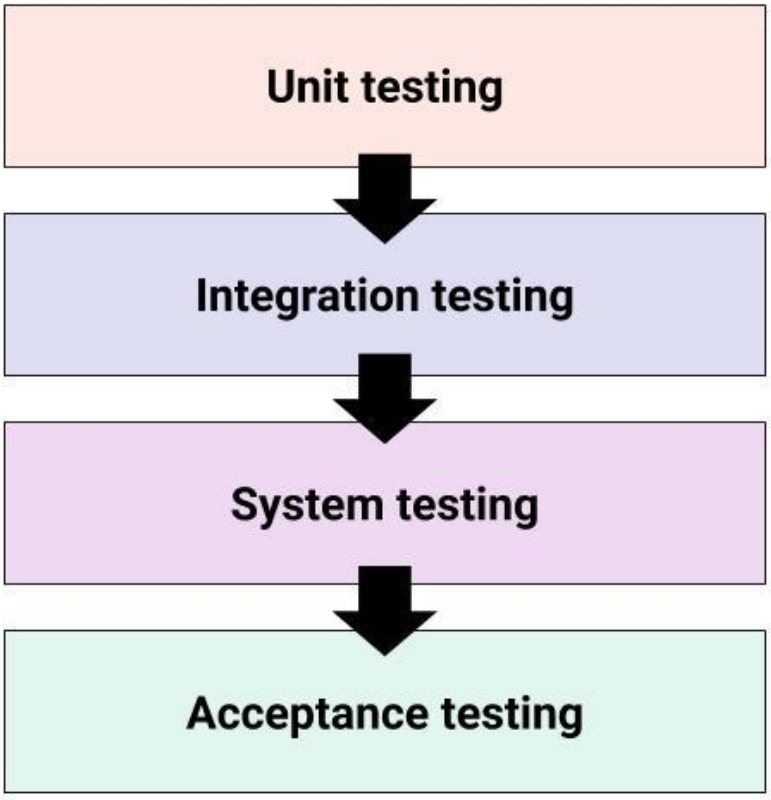


Figure 3. Waterfall diagram illustrating the software testing levels.

Unit testing

Unit, module or component testing is a method, where a software component is given some input and the created output is recorded (Juvonen 2018, 27). Usually component testing is done on components that can be tested in isolation from the system or other

components. Defects are often fixed directly as these tests can be executed by developers themselves, often times automated. (Homès 2013, 60).

Integration testing

Even though the quality of single units is necessary, it cannot guarantee the quality of the entire system, where component interact and depend on each other. In integration testing the components are combined and failures, such as a miscommunication between interfaces, can be discovered (Juvonen 2018, 27). The types of integration testing can vary a lot, as a component can be integrated in many ways. One could for example combine all components of a system at once, which might however result in trouble identifying the root cause of an issue, which can be easier with a smaller amount of component being integrated at once (Homès 2013, 61).

System testing

In system testing attributes and functioning in the testing environment of the whole system is tested. This can mean for instance the testing of a feature that requires both the frontend and backend. Failures discovered during system testing can lead to significant bug fixes and changes in the software. (Juvonen 2018, 27.) According to Homès (2013, 63-64) system testing is usually done by an independent test person or team and performs transactions from end-to-end, including all stages a user would encounter.

Acceptance testing

Once tests on lower levels of testing have been performed, acceptance testing is used to reach a level of confidence in the software. The testing environment is ideally very close to the production environment and any larger failures should have been discovered in previous tests, failures and defects can however be a byproduct of acceptance testing. Acceptance testing is usually done by the customer or user representatives, but can also be done by independent testers. The idea is however that acceptance testing is not done within the development team. (Juvonen 2018, 28; Homès 2013, 64.)

In practice not all levels of testing might be performed, but often the four levels of testing can be identified if systematic testing is part of the software development. The testing is performed by developers, within the development team and outside the development team, with the goal to identify failures early and ensure that the software meets its requirements.

2.4.2 Testing methods

Within the previously described levels of testing different testing methods or approaches can be applied. These methods differ in how they are designed and can be based on software specifications, software structure or experience and knowledge of the tester (Homès 2013, 143).

Black-box testing

In the black-box testing method the software is considered as a black-box and its contents are irrelevant, it is only tested against the requirements. This means that the code a software consists of is not observed, but usually some input is given through the software interface and verified if the output meets the requirements. (Juvonen 2018, 29; Homès 2013, 143-144.)

White-box testing

White-box testing, sometimes referred to as glass-box testing, is a structure-based approach and observes the structural components of a software. This testing method requires a more in-depth understanding of how the software works and can sometimes even mean inspecting the source code. White-box testing functions around the assumption that, if the structural components have been verified the whole system should work. (Juvonen 2018, 29; Homès 2013, 144.)

Grey-box

When applying both, aspects of black- and white-box testing, the term grey-box testing is being used. This requires some understanding of the software structure from the tester, which enables applying black- or white-box testing to different areas of the software, depending on the level of confidence necessary for these areas. (Juvonen 2018, 29; Homès 2013, 144.)

Experience testing

If the testing is not based on the specifications or structure of the software, but on the experience and knowledge of the tester, the term experience testing is used. The experience can mean experience in the business area or software testing experience in general. Experience testing is not strictly designed and leavers the tester to use their creativity and imagination and for instance let them test the software by exploring it. Even if this testing method is less defined and planned the importance of documentation is still high, so that any observations can be benefited from. (Juvonen 2018, 30-31; Homès 2013, 145.)

Static testing

All previously mentioned testing methods are dynamic testing methods, as they require some level of execution of the software in question. Testing that does not require execution of the software is called static testing. Static testing is often cheaper than dynamic testing, as defects can be discovered earlier, and fixes are there for cheaper. Examples of static testing are code reviews or static code analysis. Some defects can only be identified through static testing, but some can be identified with both, static and dynamic testing. In a peer code review for instance failures in naming conventions or internal documentation, but also issues that would result in a failure noticeable to a user, such as a missing HTML attribute, can be identified within the code. (Homès 2013, 91-94.)

3 DATA COLLECTION AND ANALYSIS

Existing literature gives a good understanding of what accessibility evaluation methodologies exist, how they are used by monitoring bodies and what software testing methodologies are commonly available. Even though testing might be partly or completely corresponding with software testing theories, the practice can be quite different from the theory, depending on the organization and development team. To identify accessibility testing methodologies that can be used in practice, not only in theory, I chose to collect data on how testing is being done from a sample of developers in teams that are affected by accessibility legislation.

3.1 Data collection

3.1.1 Interview themes

To create a natural dialog, a semi-structured interview approach was chosen. This approach does not require a list of questions, but it can be helpful to have an array of questions around the themes of the interview to make use of if necessary. The following themes were addressed in the interviews:

- Work experience with public sector clients and projects
- Testing experience and practices
    - during development
    - upon delivery
    - after delivery
- Web accessibility testing experience and practices

The outline of themes and questions can be seen in Appendix 1, but not all questions were used in all interviews and the dialog evolved around the themes with additional questions that are not part of the initial outline of questions. Depending on the strongest common language of the interviewer and the interviewee, the interview was conducted in either English or Finnish. See Appendix 2 for a translated Finnish version of the themes and questions.

The addressed themes were chosen to firstly get an understanding of the participants experience with public sector clients, which this research is mainly focusing on, as well as frame the mindset in the interview around experiences around public sector clients.

Secondly, testing experiences and practices were discussed to gather an understanding of what testing methodologies are currently used and how they are applied in web development. Lastly the topic of web accessibility testing was addressed, to gather insights on existing accessibility testing practices and possibilities.

3.1.2 Interview participants

When contacting local publicly funded organizations which had public web services, I quickly found that the development of said services is normally not being done within the organization, but outsourced by public procurement to private sector companies. These private sector companies are most often web agencies or web consultancies that offer design, development and maintenance of websites, whereas content authoring is normally being done by the organization itself. This means that developers and testers of web agencies which have been contracted by public bodies should be able to expose knowledge about any testing practices that have been applied for these public sector clients.

Finding matching candidates to be interviewed was not easy, but luckily the agency I am working for had two matching candidates which fit the criteria of having experience with public sector clients including such with accessibility requirements. After being granted permission by the company to perform the interviews, I made sure to contact colleagues that were not part of the team I have been working with, as to gather new insights and have as little bias as possible considering the circumstances. Another benefit of having participants from the same company was that we were bound by the same non-disclosure agreement which enabled the interview to be help more freely than without such an agreement. It is however good to note that both developers were working for the same agency, which is likely to reduce the variety of the gathered data.

In the scope of this bachelor's thesis and the specificity of the studied phenomenon I concluded that the sample size of two interview participants was enough to gather a good amount of data to discuss accessibility testing opportunities, but it's good to keep in mind that this dataset is not sufficient to draw any wider conclusions on testing practices in web development or accessibility testing in general.

3.1.3 Interview execution

The two selected participants were interviewed in May 2019. The interviews were held using the telecommunications software Google Hangouts as we were located in different cities and were recorded using a digital voice recorder. One interview was held in Finnish

and the other in English, as those were the languages the participants reported to be most comfortable with. The recordings of both interviews were used to manually crate transcripts, not making use of automatic transcription services. To protect the privacy of the interview participants as well as all information covered by a non-disclosure agreement, I chose not to publicly share the transcripts in this report.

3.2 Data analysis

Upon conducting the interviews and transcribing them I started the analysis by tagging the data using tags which were based on the previously defined conceptual framework, mainly around software testing levels and methods. To assist the tagging of the interview transcriptions the open-source qualitative research tool Taguette (Rampin 2020) was used.

3.2.1 Software testing in web development

The following tags in table 1 were created based on the software testing topics covered in the conceptual framework of this study.

Software testing levels	Software testing methods
• unit • integration • system • acceptance	• experience • grey-box • black-box • white-box • static • non-automated • automated

Table 1. Software testing method and level tags.

When analyzing the data, it became apparent that a mixture of software testing methods and testing practices which can be split to abstractions of software testing levels were indeed applied in web development as well. When looking at the occurrence of tags that I

added to classify testing levels and methods, it became apparent that some types of testing levels and methods were mentioned more than others. As the tag cloud in figure 4 illustrates, the most common testing level was integration testing and most common testing methods were experience and non-automated testing.

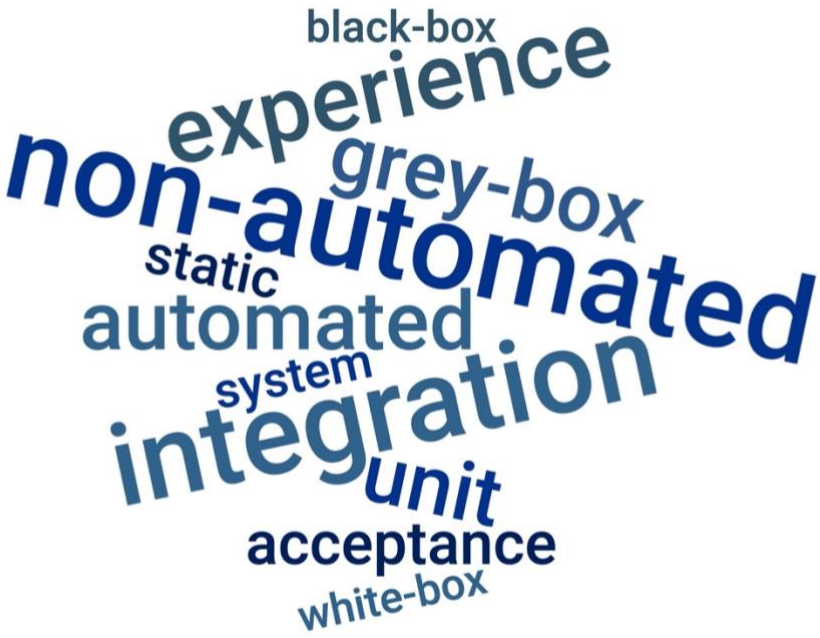


Figure 4. Tag cloud indicating occurrence of software testing levels and methods in web development.

Unit testing

Testing on the level of a unit or component was described to be performed as both, automated and non-automated testing. Automated testing on a unit level would happen through a tool running in the command line interface, which would run immediately upon saving a set of code changes and report failures. The tools were described as checking for errors in the syntax of all common programming languages, which can be classified as a static testing method. One example mentioned by an interview participant, was the following:

> *The tests are always running in the background and nowadays you can't even commit your code to version control if they don't pass.*

In addition, testing libraries were reported to be in use in some projects. These libraries would perform tests which were specifically written for a single component and checked for the correct output for a given input. These would run either upon saving, committing a change to version control or in a cloud environment before merging the change with the rest of the application. This type of functional testing can be classified black-box testing.

Non-automated testing practices on the component level were described as testing the user interface in the browser by testing if actions can be performed as expected, if the visuals match the design and that the layout is responsive, in other words if it works in all screen sizes. The extend of what and how often this is tested, it was described to be in the discretion of the developer and can be classified as experience testing, as the developer would base the decision on the testing scope on their prior experience. As the developer as tester himself has an understanding of the software structure and also checks for outputs in the user interface this type of testing can be qualified as grey-box testing.

Integration testing

When looking at integration testing, where multiple units are combined and effects between components are tested, again both automated and non-automated techniques can be identified. Automated testing methods were reported to be much less common in integration testing, however some automated visual integration test libraries are available and used to monitor for failures. These automated integration testing practices are more resource intensive and are performed later and less often than unit tests, mostly before merging a set of changes to the rest of the application, oftentimes running in a cloud environment.

Based on the experience of the interview participants it was much more common to perform integration testing in a non-automated manner, where the developer would upon completion of a component check that the styles match the design specification of that component while being used within or next to other components of the application and the entire set of components has correct styling in different screen sizes. The functionality of features where data is shared or sent between components is also tested in the user interface. This type of testing can again be classified as grey-box and experience testing. Depending on the team structure, similar integration testing is also performed by other team members or dedicated testers, most of the time in dedicated testing environments. In this case black-box testing is most likely in question, as the tester does normally not have insights about the software structure, but is simply focusing on the inputs and outputs of the user interface.

At the same stage as possible automated integration tests are performed, the step before merging code changes with the existing code, the code was also reported to be reviewed by other developers, which is called performing a code review or peer review. Depending on the entity size of the code changes this type of static testing can be classified as either integration or unit testing, as it focuses on a complete set of changes about to be made available to other team members or the customer through a dedicated testing environment. The code review covers review the syntax, but can depending on the reviewer and the particular development item also include more in-depth reviews covering the functionality or general best and common practices among the development team.

Before proceeding to further testing or the release of the code change, interview participants reported that in addition to visual and functional testing, the developed items are also tested in different web browsers. This browser testing means that the visuals and functionalities verified to be compatible with all browsers that were previously agreed to be supported or at least the more commonly used browsers. A participant described their browser testing practice as follows:

> *I probably build the site 99% done and then I check browsers, because there's no point in me tweaking if I'm still checking stuff with the client, or they need to approve something.*

System testing

Once the development proceeded to a stage where it has passed unit and integration testing it sometimes undergoes testing practices that can be classified as system testing. The testers, often another team member or a dedicated tester, would test user flows that include updated components, often end-to-end so including all stages a user would encounter. Depending on the project the tester would at this stage also do browser testing, similar to the one done as part of the integration testing. This type of testing is mostly performed in a non-automated manner and as black-box testing, as the tester usually is not aware of the software structure.

According to the interview participants this type of extensive system testing is not very common in their own experience.

Acceptance testing

Testing that can be classified as acceptance testing is most commonly performed by the customer who ordered the development item. This type of testing was described as sign-off testing where the customer would validate that the requested changes were done and are working, after which the change would be released to production. The changes are

normally presented in a testing environment to the customer first, to enable this type of testing easily. As the customer has regularly no insight about the code, this type of testing is qualifiable as black-box testing. One interview participant described this testing practice as follows:

> *You have like the client tests of course. They sort of like validate what you've done, like is this good, and they put their checkmark, like: "Yeah, this is what I want, this is all good".*

After the release of a code change it would also occur that end users would encounter failures, for instance while using a browser that was not tested, which would then result in the customer reporting the end user feedback to the development team and them fixing the issue.

Testing level and method combinations

After the data from the interviews was tagged and different software testing levels and methods identified it was possible to further analyze the data by showing which testing methods were used at which levels, as table 2 shows.

	Unit testing	Integration testing	System testing	Acceptance testing
Automated	YES	YES	NO	NO
Non-automated	YES	YES	YES	YES
Static	YES	YES	NO	NO
Experience	YES	YES	NO	NO
Black-box	YES	YES	YES	YES
White-box	NO	NO	NO	NO
Grey-box	YES	YES	NO	NO

Table 2. Matrix of testing methods usage at different testing levels.

The classification if a method was used at a certain testing level was based on the reported prevalence of that testing method by the interviewees, but it is important to keep in mind that depending on the type of project and development team the chosen methods

might differ greatly. The reported differences of used testing practices are also substantial and it should be kept in mind that testing practices cannot be generalized and applied blindly, neither for software testing nor for web development testing specifically.

3.2.2 Accessibility testing techniques in web development

As the chosen interview participants already had some experience with projects that required accessibility standard fulfillment, they were able to give insights on some accessibility testing practices being applied. To be able to identify accessible testing practices as well as being able to classify them as software levels and methods, the same techniques of applying the previously defined tags was used. Several testing techniques were reported to have been used.

Automated accessibility testing library

Similar to automated testing tools that evaluate code for correct syntax in their respective languages, tools which automatically test for accessibility failures are available. These tools were reported to be running on the command line interface, automatically validating code upon saving and informing the developer immediately about possible failures. Similar to automated tools validating the syntax, these accessibility testing tools can be classified as static testing methods and are mostly operating on the unit level. These testing tools were reported to be quite popular by developers, as they inform about failures at the earliest stage possible without additional manual testing effort.

Keyboard testing

The developers reported that during development they would, in addition to functional testing by using the mouse in the user interface, from time to time test the functionality by only using the keyboard for navigation in the browser and website, in an effort to find possible accessibility failures. This type of testing would happen relatively early and often and could be classified as either unit, integration or system testing, depending on the phase of the development, as it was reported to be used at most phases in the development, similar to performing functional testing with the mouse in the user interface. It was reported that keyboard testing was often applied based on the developer's evaluation if the set of changes benefit from additional testing with the keyboard. When performed by the developer itself, this technique would fall under the grey-box testing method and possibly experience testing, but can also be classified as black-box testing when being performed by someone else.

Screen reader testing

Only after almost the complete development of an entity was finished, developers reported to be using a screen reader to discover possible issues that screen reader users might encounter. Screen readers were described as assistive technology that is used by visually impaired users to read out text and other visual content on a web site. The usage of a screen reader was described to being done at around the same time and interval when browser testing was done, where compatibility with different devices and software was ensured. As this type of testing was described as somewhat bothersome and requiring special knowledge on using the screen reader, it was only applied whenever the developer saw it necessary, which classifies it as a type of experience testing, performed at the integration testing level. Depending on the project, multiple development items would be gathered and tested in a later stage, closer to the release of a large set of changes, where end-to-end tests were also performed by using a screen reader. In this context the screen reader testing also happens on the system level. Depending on the tester this can be considered as either black-box or grey-box testing.

Accessibility testing browser extension

At a similar stage and frequency as using the screen reader, the usage of accessibility evaluation tools as browser extensions was reported. These were however described to report partly similar failures that automated tools running in the command line would be able to find. Additional features are however the ability to find issues that only become apparent when combining multiple components or modules, where the browser extension was able to identify failures which the command line tool could not. The tools were described as multi-functional and their separate functions would sometimes only be used when deemed necessary by the developer, which can qualify this type of testing as experience testing. As the browser extension only evaluates one page as a time they can be classified as being part of integration level testing. The tool is used on a visible and interactive user interface, but can at the same time still analyze parts of the code that are made available in the browser, which qualifies as grey-box testing.

Code review

When a peer review covering general syntax and code quality failures is performed, it was reported that accessibility considerations were concurrently also commonly addressed, depending on the experience of the reviewer. The review covered however mostly failures which can by their nature be seen from the code without executing it. This type of static testing can either fall under the unit or integration testing level, depending on the entity size of the development item. Code review could qualify as experience testing, as it is only

possible to discover accessibility failures if the reviewer has prior accessibility development and testing experience.

External audit

As the customers are often due to lack of expertise not able to verify if the requirements of meeting accessibility standards are met, it was reported that using third parties to perform an audit of a delivered site or application was common. These companies would have experts testing the site and its user-flows using similar techniques as previously mentioned, but often more extensive and sometimes with actually disabled users. This type of acceptance testing was reported to be done in longer intervals and could be qualified as experience and black-box testing.

3.2.3 Accessibility testing techniques at software testing levels and methods

The data shows that there are several software testing methods used at most or all software testing levels in web development. Also, many of the reported accessibility testing techniques fit into the forementioned categories and can be mapped to show which technique was used at what level.

	Unit testing	Integration testing	System testing	Acceptance testing
Automated accessibility testing library	YES	NO	NO	NO
Keyboard testing	YES	YES	YES	NO
Screen reader testing	NO	YES	YES	NO
Accessibility testing browser extension	NO	YES	NO	NO
Code review	YES	YES	NO	NO
External audit	NO	NO	NO	YES

Table 3. Matrix of testing technique usage at software testing levels.

As seen in table 3, all testing levels have testing techniques available that were already reported to be in use in many projects requiring accessibility standard compliance. At the unit testing level, the use of automated testing libraries, keyboard testing as well as having the code reviewed by peers are prevalent. During integration testing keyboard testing, screen reader testing, accessibility testing browser extensions as well as code review are available. For system testing purposes the number of techniques is down to keyboard testing and screen reader testing, whereas for acceptance testing only the use of external audits is left.

	Automated	Non-automated	Static	Experience	Black-box	Grey-box
Automated accessibility testing library	YES	NO	YES	NO	NO	NO
Keyboard testing	NO	YES	NO	YES	YES	YES
Screen reader testing	NO	YES	NO	YES	YES	YES
Accessibility testing browser extension	YES	YES	YES	YES	NO	YES
Code review	NO	YES	YES	YES	NO	NO
External audit	YES	YES	YES	YES	YES	YES

Table 4. Matrix of testing technique classified as software testing methods.

When trying to map accessibility testing techniques to software testing methods, as seen in table 4, it becomes apparent that all software testing methods have multiple available techniques to choose from, which have already been applied successfully. There are some automated tools, such as testing libraries and browser extensions or companies

providing automated audits, but the non-automated techniques still seem to be more common when aiming for accessibility compliance. Non-automated techniques include doing keyboard testing, screen reader testing, evaluation with the help of browser extensions as well as code reviews. Available static testing methods are the usage of testing libraries, browser extensions as well as code reviews and possibly external audits. When looking at experience testing, almost all techniques and tools are available to the tester and they can mix and match those depending on the situation, with the exception of testing libraries which have to be set up and used in the development phase. For the black-box testing method keyboard and screen reader testing techniques are available, which external audits can also make use of. Grey-box testing can cover the same aspects as the previously mentioned techniques but can additionally include the use of browser testing extension.

3.2.4 Accessibility testing practices and evaluation method recommendations

As discussed in the accessibility evaluation section of this report, the WAI has lined out methodologies and recommendations on how to evaluate WCAG conformance. The WCAG-EM itself is meant to evaluate an existing website by auditing a representative sample and is not meant as an approach to testing in feature development. Due to this the method itself, or equivalent practices, were not mentioned by the interviewed developers, but it might be that the mentioned external audits that were used for acceptance testing would make use of this methodology as it was designed for such a use-case.

The WAI also suggested the use of evaluation tools, out of which command line tools as well as browser plugins were reported to have been in use by the interview participants. As these tools are only meant to be used as a supporting tool in addition to non-automated testing, as only some WCAG success criteria can be tested with evaluation tools. The interviewed developers reported that the forementioned tools indeed only made up a part of all accessibility testing practices.

Non-automated testing practices are still necessary according to the WAI, even though efforts are made to further develop and automate testing tools to cover more WCAG success criteria. The interviewed developers reported as well that non-automated testing is used during feature development to find possible WCAG success criteria failures and it was deemed necessary and useful.

When performing accessibility testing, a study has found expert testing to be much more efficient than nonexpert testing. It is arguable when a tester can be defined as nonexpert or expert, but the interviewed developers clearly had some experience on which they

already relied upon when making a decision on which accessibility technique they would apply and when or what interval they would do so. They also reported to rely upon experienced peers for performing code reviews, as well as external audits being used to get more accessibility expert testing and achieve a high confidence to be WCAG conformant.

When reviewing the plan of the EU to monitor conformance of the EU directive, it is noticeable that for the larger scale simplified monitoring evaluation tools are used to get a general overview of WCAG conformance concerning automatically testable success criteria. Moreover, in-depth monitoring also includes non-automated expert testing to gather a more detailed overview of complete WCAG conformance for a smaller subset of monitored pages. If the EU directive monitoring is seen as a guideline to monitoring bodies of each EU country, it could be argued that website administrators applying at least the same evaluation methods should lead to a sufficient level of conformance.

4 CONCLUSIONS

The research question on how web accessibility testing methods could be integrated within existing web development testing practices can be answered in short when stating that by identifying currently used testing practices it is possible to find and use accessibility testing practices that are performed at a similar testing level using similar testing methods.

When aiming for WCAG conformance in existing websites, methodologies such as the WCAG-EM and using external audits might be a good starting point. If the aim however is to further develop a website or even develop a completely new website, accessibility testing techniques could be applied – same as other forms of software testing – early and often.

These forementioned testing techniques could, based on my research, be chosen and applied based on other similar software testing techniques. To find matching accessibility testing techniques it can be useful to analyze what software testing methods are already applied at what testing level and choose an accessibility testing technique that works with the same method and level. My research supports the thesis of existing software testing practices in web development and existing accessibility testing in web development having some overlap. This could ease the integration of new accessibility testing routines and practices in pre-existing workflows and practices. For instance, when the developing team is making use of testing tools to perform static code analysis on a unit level that runs in the command line it would be a good opportunity to make use of accessibility testing libraries that perform similarly static code analysis on a unit level. Another example of a testing opportunity could be grey-box integration testing that is performed by a developer once a component is finished and about to be introduced to the codebase. Here additional grey-box integration accessibility testing practices, such as keyboard testing, screen reader testing as well as making use of an accessibility testing browser extension could be applied in addition to the preexisting testing practices.

Other research has shown that the use of experts and performing expert accessibility testing is highly efficient when compared to testing by nonexperts, as experts were able to identify more possible failures with less time spent on testing. This could advocate for training team members that already perform some testing to subject matter experts on accessibility testing, which would help identify more barriers and spread knowledge to other team members on what considerations could be taken into account.

5 DISCUSSION

My research has shown that existing testing practices in web development have many similarities with available accessibility testing practices. When classifying both using software testing methods and testing levels, it is easy to notice with which techniques and practices there is an overlap between the two. Using this type of systematic approach to choose and apply accessibility testing methods might prove useful to development teams aiming for WCAG conformance or being accessibility in general.

5.1 Industry recommendations

Development teams that are aiming for accessibility or WCAG conformance in the web development industry should make themselves aware of the available accessibility testing practices. By then looking at their existing testing practices, possibly by classifying them to software testing concepts such as testing methods and levels, it could prove effective to choose and apply accessibility testing methods that work under similar conditions, so using the same methods and levels. This could make the integration of new testing practices smoother and might not slow down testing or development unnecessarily.

In addition, other research has shown that using experts for accessibility testing is much more efficient than using nonexperts, so it might be useful to either hire or train one or more testers to become accessibility testing experts.

5.2 Research recommendations

My research was able to identify overlaps between existing testing practices in web development and accessibility testing practices. Based on these findings it could be argued that choosing and applying these accessibility testing practices in areas where similar testing is performed could be most effective. Researching that very question – if applying these testing practices in the forementioned manner actually is most effective – could be the next step, in an effort to establish how to efficiently and effectively test for accessibility.

As accessibility testing practices are being adapted on a larger scale due to recently passed legislations, it could also be interesting to further research what accessibility testing practices are used in websites and their developing teams. These testing practices could then be compared regarding their efficiency in achieving a high level of accessibility compared to time spent on testing and fixing accessibility failures. These findings could further validate or invalidate the results of this research.

REFERENCES

Abou-Zahra, S. 2017a. Diversity of Web Users. W3C. [Cited 29 October 2017]. Available at: https://www.w3.org/WAI/intro/people-use-web/diversity

Abou-Zahra, S. 2017b. Selecting Web Accessibility Evaluation Tools. W3C. [Cited 24 April 2019]. Available at: https://www.w3.org/WAI/test-evaluate/tools/selecting/

Aluehallintovirasto. 2019. Saavutettavuusvaatimukset ja WCAG. [Cited 3 April 2019]. Available at: https://www.avi.fi/web/avi/saavutettavuus-ja-wcag

Baresi, L. and Pezzè, M. 2006. An Introduction to Software Testing. Electronic Notes in Theoretical Computer Science. Vol. 148 (1), 89-111.

Brajnik, G., Yesilada, Y. and Harper, S. 2011. The Expertise Effect on Web Accessibility Evaluation Methods. Human-Computer Interaction. Vol. 26 (3).

Brewer, S. 2002. Using Combined Expertise to Evaluate Web Accessibility. W3C. [Cited 24 April 2019]. Available at: https://www.w3.org/WAI/test-evaluate/combined-expertise/

Caldwell, B., Reid, L. G., Vanderheiden, G., Chisholm, W., Slatin, J. and White, J. 2008. Web Content Accessibility Guidelines (WCAG) 2.0. W3C. [Cited 29 October 2017]. Available at: https://www.w3.org/TR/WCAG20/

Clark, J. 2003. Understanding Web accessibility. Journal of Volunteer Administration. Association for Volunteer Administration, Vol. 21 (1), 36-39. [Cited 8 May 2019]. Available at: https://joeclark.org/access/webaccess/JVoluntAdmin.html

Commission Implementing Decision (EU) 2018/1524. 2018. EUR-Lex. Official Journal of the European Union. [Cited 23 April 2019]. Available at: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32018D1524&from=EN

Cooper, M., Kirkpatrick, A. and O Connor, J. 2016. Understanding WCAG 2.0. W3C. [Cited 29 October 2017]. Available at: https://www.w3.org/TR/UNDERSTANDING-WCAG20/complete.html

Dardailler, D. 2009. WAI early days, Web Accessibility Initiative (WAI). W3C. [Cited 9 April 2019]. Available at: https://www.w3.org/WAI/history

European Commission. 2018. Digital Single Market. Web Accessibility. [Cited 16 April 2019]. Available at: https://ec.europa.eu/digital-single-market/en/web-accessibility

Friedman, M. G. and Bryen, D. N. 2007. Web accessibility design recommendations for people with cognitive disabilities. Technology & Disability. IOS Press, Vol. 19 (4), 205–212.

Henry, S. L. 2005. Introduction to Web Accessibility, Web Accessibility Initiative (WAI). W3C. [Cited 29 October 2017]. Available at: https://www.w3.org/WAI/intro/accessibility.php

Henry, S. L. 2018. Web Content Accessibility Guidelines (WCAG) Overview, Web Accessibility Initiative (WAI). W3C. [Cited 9 April 2019]. Available at: https://www.w3.org/WAI/standards-guidelines/wcag/

Henry, S. L. and Abou-Zahra, S. 2016. WCAG-EM Overview: Website Accessibility Conformance Evaluation Methodology. W3C. [Cited 16 April 2019]. Available at: https://www.w3.org/WAI/test-evaluate/conformance/wcag-em/

Homès, B. 2013. Fundamentals of Software Testing. London: Wiley.

Juvonen, R. 2018. Ohlemistoprojektin sudekuopat ja miten ne vältetään. Helsinki: Books on Demand.

Kirkpatrick, A., O Connor, J., Campbell, A. and Cooper, M. 2018. Web Content Accessibility Guidelines (WCAG) 2.1. W3C. [Cited 11 April 2019]. Available at: https://www.w3.org/TR/WCAG21/

Laki digitaalisten palvelujen tarjoamisesta 306/2019. Finlex. [Cited 3 April 2019]. Available at: https://www.finlex.fi/fi/laki/alkup/2019/20190306

Laurin, S., Cederbom, A., Martinez-Usero, J., Kubitschke, L., Moledo, A., Simons, B. and Abou-Zahra, S. 2016. Monitoring methodologies for web accessibility in the European Union. European Union. [Cited 9 April 2019]. Available at: http://ec.europa.eu/newsroom/dae/document.cfm?doc_id=19274

Logacheva, E. 2016. Designing for Web Usability and Accessibility: User-Interface Design Guidelines in Connection with Human-Computer Interaction. Bachelor's thesis. Helsinki: Metropolia University of Applied Sciences. [Cited 29 October 2017]. Available at: http://urn.fi/URN:NBN:fi:amk-201605066388

Mitronen, M. 2018. EU:n saavutettavuusdirektiivin vaikutukset verkkosivustoihin. Bachelor's thesis. Haaga-Helia University of Applied Sciences. [Cited 9 April 2019]. Available at: http://urn.fi/URN:NBN:fi:amk-2018122022604

Mueller, M. J., Jolly, J. and Eggert, E. 2018. Web Accessibility Laws & Policies, Web Accessibility Initiative (WAI). W3C. [Cited 9 April 2019]. Available at: https://www.w3.org/WAI/policies

Ng, C. 2017. A Practical Guide to Improving Web Accessibility. Weave: Journal of Library User Experience. Michigan Publishing, University of Michigan Library, Vol. 1 (7).

Online English Dictionary 2017. [Cited 29 October 2017]. Available at: https://en.oxforddictionaries.com/

Rampin, R., Steeves, V. and DeMott, S. 2020. Taguette (Version 0.9.2). Zenodo. [Cited 10 September 2020]. Available at: https://doi.org/10.5281/zenodo.4002742

Riley-Huff, D. A. 2012. Web Accessibility and Universal Design. Library Technology Reports. American Library Association, Vol. 48 (7), 29-35.

Rowland, D.R. 2014. Reviewing the Literature: A Short Guide for Research Students. The University of Queensland. [Cited 29 April 2019]. Available at: https://uq.edu.au/student-services/pdf/learning/lit-reviews-for-rx-students-v7.pdf

Saaranen-Kauppinen, A. and Puusniekka, A. 2009. Menetelmäopetuksen tietovaranto KvaliMOTV. Toinen vedos. Tampere: Yhteiskuntatieteellinen tietoarkisto. [Cited 29 April 2019]. Available at: https://www.fsd.uta.fi/fi/tietoarkisto/julkaisut/kvalimotv.pdf

Sargeant, J. 2012. Qualitative Research Part II: Participants, Analysis, and Quality Assurance. Journal of graduate medical education, Vol. 4 (1), 1-3. [Cited 8 May 2019]. Available at: https://dx.doi.org/10.4300%2FJGME-D-11-00307.1

Velleman, E. and Abou-Zahra, S. 2014. Website Accessibility Conformance Evaluation Methodology (WCAG-EM) 1.0. W3C. [Cited 16 April 2019]. Available at: https://www.w3.org/TR/WCAG-EM/

W3C 2017. Web Accessibility Initiative (WAI), Web Accessibility Initiative (WAI). [Cited 29 October 2017]. Available at: https://www.w3.org/WAI/

W3C 2019. WAI-Tools. [Cited 24 April 2019]. Available at: https://www.w3.org/WAI/about/projects/wai-tools/

APPENDICES

Appendix 1. Interview themes and questions

1) Work experience with public sector clients and projects
    a) Tell me about your experience working with public sector clients.
    b) Describe your public sector projects and their scope.
2) Testing:
    a) In general
        i) What do you understand by the term testing in web development?
        ii) Would you like to test more or less?
        iii) What type of testing is very important? What kind could be left away?
        iv) What type of testing feels very efficient? What kind less efficient?
    b) During development
        i) How do developers test or check that the code works?
        ii) When and how often is testing done?
        iii) What techniques or tools do you use for testing?
        iv) How do other team members perform complementary testing?
    c) Upon delivery
        i) How and by whom is the software tested or checked before it is delivered?
        ii) How are bugs and issues revealed during this phase addressed?
    d) After delivery
        i) How and by whom is the software tested after it is delivered or deployed?
        ii) How are bugs and issues revealed during this phase addressed?
3) Web accessibility
    a) What do you understand by the term web accessibility?
    b) If you are already testing for accessibility, how and by whom is accessibility testing done?
    c) Where could you improve on accessibility testing?

Appendix 2. Interview themes and questions (translated from English to Finnish by Oksi)

1) Työkokemus julkisen sektorin asiakkaiden ja projektien parissa
   a) Kerro kokemuksistasi julkisen sektorin asiakkaiden kanssa.
   b) Kuvaile julkisen sektorin projekteja ja niiden laajuutta.
2) Testaaminen:
   a) Yleisesti
      i) Mitä ymmärrät termillä testaaminen verkkokehityksessä?
      ii) Haluaisitko testata enemmän tai vähemmän kuin nykyään?
      iii) Minkä tyyppinen testaaminen on erittäin tärkeä? Minkä voisi jättää pois?
      iv) Minkä tyyppinen testaaminen tuntuu tehokkaalta? Minkä tyyppinen vähemmän tehokkaammalta?
   b) Kehityksen aikana
      i) Miten kehittäjät testaavat tai tarkistaa että koodi toimii?
      ii) Kuinka usein ja miten testaaminen suoritetaan?
      iii) Mitä työkaluja tai menetelmiä käytä testaamiseen?
      iv) Miten muut tiimin jäsenet täydentävät testaamista?
   c) Toimituksen yhteydessä
      i) Miten ja kenen toimesta ohjelmistoa testataan tai tarkistetaan ennen kuin se toimitetaan?
      ii) Miten käsitellään bugit ja ongelmat, jotka ilmenevät tässä vaiheessa?
   d) Toimiuksen jälkeen
      i) Miten ja kenen toimesta ohjelmistoa testataan sen toimituksen tai julkaisemisen jälkeen?
      ii) Miten käsitellään bugit ja ongelmat, jotka ilmenevät tässä vaiheessa?
3) Saavutettavuus verkossa
   a) Mitä ymmärrät termillä saavutettavuus verkossa?
   b) Jos testaat saavutettavuutta, miten ja kenen toimesta saavutettavuustestaus suoritetaan?
   c) Missä voisit parantaa saavutettavuustestausta?