Empowering Agile Project Members with Accessibility Testing Tools: A Case Study

Viktoria Stray[1(✉)], Aleksander Bai[2], Nikolai Sverdrup[1], and Heidi Mork[3]

[1] Department of Informatics, University of Oslo, Oslo, Norway
{stray, njsverdr}@ifi.uio.no
[2] Norwegian Computing Center, Oslo, Norway
aleksander.bai@nr.no
[3] NRK, Oslo, Norway
heidi.mork@nrk.no

Abstract. There is a growing interest in making software more accessible for everyone, which is emphasized by the numerous suggestions passed into law in many countries. However, many software organizations that use agile methods postpone or neglect accessibility testing. We aimed to understand how accessibility testing can be better integrated into the daily routine of agile projects by conducting a case study in a Norwegian software company. We investigated three accessibility testing tools: automatic checker, simulation glasses, and a dyslexia simulator. We hosted sessions at which agile project members used the tools while thinking out loud, responded to questionnaires, and were interviewed at the end. Additionally, we observed the project members for 18 workdays. Our results show that all three tools are suitable for agile projects. Especially the automatic checker and simulation glasses worked well in finding accessibility issues and were described as easy to use by the project members. Software organizations should empower their agile project members with low-cost and efficient accessibility testing tools to make their products more accessible for all. Doing this early and often in the development cycle may save the project from potential high costs at a later stage.

Keywords: Accessibility testing · Usability · Cambridge simulation glasses · SiteImprove accessibility checker · WCAG · Agile software development · Universal design

1 Introduction

Creating software that is accessible for everyone (including users with impairments) is an important consideration for an increasingly digital society. Accessibility focuses on enabling people with the widest range of capabilities to use a product or removing barriers that can interfere with the user experience [1]. However, putting accessibility into practice remains a challenge in software development [2], and agile methods are especially under scrutiny for not offering enough consideration for accessibility [3, 4].

Testing for accessibility means testing how users with impairments and disabilities (e.g., dyslexia or impaired vision) will experience the software product. Although agile

P. Kruchten et al. (Eds.): XP 2019, LNBIP 355, pp. 86–100, 2019.
https://doi.org/10.1007/978-3-030-19034-7_6

methods highlight principles such as delivering working software and regular testing, incorporating accessibility testing throughout an agile process is complicated and less common [3, 4], and it can be an expensive endeavor [5]. Accessibility testing is often postponed to the late phases of software development which breaks with the principle of delivering working software frequently [6]. Researchers argue for easier methods to make finding accessibility flaws more effective in agile software projects [6, 9, 10]. Agile projects face a growing complexity with a variety of organizational constraints such as universal design, legislation, and security, forcing the cross-functional agile team to adjust their practices and experiment with new team structures (e.g., BizDev teams) [7, 8]. Thus, now might be a good time to include practices and tools that let agile teams test for accessibility issues throughout the project.

In an agile process, which features short iterative cycles, it would be difficult to fit costly and time-consuming accessibility testing into regular testing procedures. Accordingly, there is a need to find methods that integrate well with agile approaches. Methods and tools that can be utilized often without being too resource- and time-intensive will fit well with agile principles. Our motivation for this project was thus to improve accessibility in agile software development by testing different methods that are considered fast and efficient.

The benefits of having an accessible solution are apparent. Poorly developed software with regard to accessibility is aggravating to impaired users and can make the product miss out on potential users, giving the issue both social justice and economic motivation. Research has also shown that software that accommodates accessibility and inclusion will benefit all users, including those without any impairments [11]. Also, for software to be considered working, it needs to be thoroughly tested and ready to be delivered to customers or production. If software fails to be sufficiently accessible, it cannot be considered to be working. For the goal of having working software, accessibility testing needs to be a part of every iteration that involves features which can affect the accessibility of the software. Motivated by this, we set out to investigate how accessibility testing can be integrated into agile software development, by answering the following research question:

RQ: How do agile project members experience using accessibility testing tools?

To address this question, we conducted a case study [12] where we had the participants test three different accessibility tools and respond to a questionnaire about the tools they tested. The participants were also interviewed and observed.

The remainder of this paper is organized as follows. Section 2 outlines the relevant background on accessibility testing in agile software development. Section 3 discusses different testing tools. Section 4 describes the data collection. Section 5 presents the results, and Sect. 6 discusses the results and implications for practice. Section 7 concludes and suggests future work.

2 Background

Failing to make a product accessible can be costly, as it will increase the probability of having to make late and expensive changes in the development process and further lead to prolonged development time. A product with which the user cannot achieve their goal through the usual procedure will also lead to a reduction in users and increase the need for additional support and assistance to help guide the users, such as help-desk services and on-site support [13]. Those who do include accessibility in their development process will find that the cost for testing procedures can take its fair share out of the development budget, especially since the requirements for accessibility testing often are complex [14].

Investing in accessibility testing early and throughout the development process can yield a substantial return on investment [15]. Detecting mistakes early lowers the cost of fixing the error and avoids the accumulation of big, time-consuming problems at the end of the development process. Accessibility testing, however, has the unfortunate fate of often being conducted at the very end of development [2]. Reduction in the amount of work needing to be done in the maintenance phase can also be a substantial benefit, as a significant portion of the life-cycle expenditure of a software product can congregate in this later stage.

2.1 Accessibility Testing in Agile Software Development

Dissatisfaction surrounding traditional accessibility test methods is a prevalent theme in other research. Researchers have investigated usability issues in agile development (e.g., [16–18]) but few researchers have investigated accessibility testing in agile projects. As Luján-Mora and Masri [9] argue, accessibility is difficult to achieve with traditional software development practices, due mainly to its habit of being implemented too late in the development process. They propose that agile development can significantly help to improve web accessibility due to its focus on regular testing.

Similarly, Eklund and Levingston [6] discuss how usability can benefit from agile development. They propose a set of steps one can use to incorporate usability techniques in agile development, such as conducting smaller tests and reviews to cover more of the development process as well as to rely extensively on external consultants with expert knowledge on accessibility testing. Meszaros and Aston [19] describe the introduction of early accessibility testing in an agile development process using testing methods such as paper prototypes to locate accessibility bugs, wizard of Oz testing, and usability stories. This resulted in a higher acceptance by end users. Kane [3] highlights the lack of accessibility testing in agile development and proposes techniques to integrate more accessibility testing into already established agile practices.

3 Testing Tools

Testing for accessibility entails testing for a multitude of different impairments and disabilities in various states. When ensuring a product works for users with eyesight-related issues, for example, one has to consider aspects such as different levels of visual

perception, color blindness, complete blindness, tunnel vision, and so on. All of these affect the user experience in different ways and create different challenges that must be solved. Without any tools, a developer would have to retain an extensive amount of knowledge to meet the demands of accessibility testing, which would be hard to put into practice. One solution is to hire expert help from accessibility consultants or recruit people with different impairments to conduct user testing. However, this is an expensive solution and should be done when the product moves into a more stable phase.

One of the most significant challenges in testing for accessibility is that one has to account for a wide range of people. Using one super-tool that can take account of everything is not realistic due to the diversity of impairments and the different ways they affect users. When testing with accessibility in mind, there is a need to evaluate everything from content, layout, comprehensibility and technical implementation to compatibility with other accessibility tools, such as a screen reader.

There are some tools that we considered but decided not to include. For example, we considered a screen reader and WCAG (Web Content Accessibility Guidelines). However, both tools are intensive regarding the knowledge required to operate them, and this does not fit very well into an agile process with short iterative cycles. A screen reader is a software tool intended mainly for use by people suffering from mildly impaired vision to complete blindness. The biggest challenge of using a screen reader is that it required extensive training in order to operate the tool correctly. It may take years to achieve a proficient use of a screen reader, and advanced users can navigate around an obstacle that novice users cannot. WCAG is the de facto practiced method used to check for accessibility. WCAG is presented as a set of specifications categorized into four broad principles containing 61 numbered paragraphs of success criteria. Each criterion has its own detailed page, specifying the intent, examples, techniques, key terms, and related resources associated with the paragraph. The techniques and success criteria will, altogether, give a total of 379 different pages, each with a description, examples, additional resources, test procedures, and expected test results. From our earlier research, we know that developers perceive WCAG as tedious, cumbersome, and time-consuming to use in agile projects [14].

We chose the testing tools SiteImprove, Cambridge simulation glasses, and a dyslexia simulator (where we developed a Chrome browser extension), briefly presented below, based on other studies that have categorized testing tools according to barrier groups and cost-benefit [14, 20]. Since we wanted testing tools that could be easily integrated into an agile process (low cost and little prior knowledge), we found these to be the best candidates.

3.1 SiteImprove Accessibility Checker

SiteImprove is an automatic checker and browser extension that can analyze a web-page for breaches of many WCAG criteria. When activated, the extension will automatically analyze the currently opened web-page for noncompliance with the different levels of the WCAG technical standard. It distinguishes between the different categories of the WCAG standard and organizes them into groups. Each instance of error can give a direct link to the WCAG manual for a more detailed explanation of why the

error exists and includes suggestions about how one can achieve compliance with WCAG. Other notable features are the ability to highlight where the error exists on the site itself and in the public source code by highlighting in the browser's developer tools.

One of the most significant drawbacks with automatic checker tools is that there are many things they cannot check, at least not until we have better artificial intelligence. For instance, an automatic checker can make sure that images have alternative HTML tags, but it cannot check if the description is accurate and meaningful.

3.2 Cambridge Simulation Glasses

A person can identify accessibility faults by wearing blurred glasses while interacting with the interface or object that is under examination. For our testing, we used the Cambridge simulation glasses. The glasses are thin enough to stack several pieces of glasses in sequential layers, enabling the tester to be in control of the degree of reduced vision. Before using the glasses, the tester performs a quick eyesight evaluation using a Snellen chart that comes with the glasses or can be printed out. The chart indicates how many glasses the tester should wear to simulate reduced vision (a 95% coverage of the general population). Most people will need two pair of glasses, but those with very good eyesight might require three glasses. While wearing the glasses, the tester interacts with whatever is to be evaluated for accessibility, and any possible shortcomings should be made readily apparent for the tester as they might struggle to use the solution.

3.3 Dyslexia Simulation

The dyslexia simulator is a small and simple browser extension that tries to simulate how a dyslectic user or someone with some other reading disorder might experience a website. Dyslexia is a common learning disability, and the condition can severely hamper a user's ability to comprehend a website. Dyslexia impedes the ability to read, and how software is presented can significantly affect the difficulty of interaction for a dyslectic user, especially in sites that feature much text or require the user to write [21]. The tool will test a website by shuffling the letters in words, making it difficult to read.

When testing a solution, the tester operates the interface as normal with the dyslexia software running; if there are areas that become difficult to perceive, the tester can use that as an indication for something that might become difficult for someone with reading difficulties to perceive. The dyslexia simulation tries to highlight issues related to a neurological impairment, which is challenging to test for. It is difficult to make software that can detect areas affected by such disabilities, and it is difficult for a tester to understand what the impaired user might experience. There are several WCAG paragraphs detailing the issues related to reading impairments and steps to help in the matter.

We used a dyslexia simulator in a pilot study that was a script the user had to rerun for each use, which was cumbersome. Therefore, for this study, we made an extension for the Google Chrome web browser (now available on the Chrome web store) that is easier to install and use.

4 Data Collection

We conducted this study in a Norwegian software company that makes digital services and solutions for banks. We observed the project members in their daily routines and during agile practices such as daily stand-up meetings [22] and retrospective meetings [23]. We documented their behavior, and conversation topics and were especially looking for matters where accessibility issues could play a part. We also interviewed four members with different roles and responsibilities. The reviews of the three testing methods were conducted by hosting sessions in which the participants tried out accessibility testing tools, answered questionnaires about the tools, and were interviewed at the end of the session, see Table 1 for an overview of the data collection.

Interview questions and testing tasks were made after determining what information was needed and how much time we could have with the participants. As recommended by [24], every tester had filled out a background survey before the session. After a pilot attempt of the testing session, it was decided to use a publicly available website not affiliated with the participants, after we found the participants' knowledge of their products to influence their ability to judge the accessibility.

All the testing sessions were originally scheduled for one hour each, where we had time to test three methods, which involved a quick briefing of the participants about the tools and instructions on how to use them. In the testing sessions, the participants were instructed to solve pre-made tasks that required them to use the tools to locate accessibility faults in a website. As the participants used the tools, they would comment on their experience and be prompted with questions by the interviewers to reflect on the experience. After completing the tasks, they filled out a questionnaire.

Having finished testing, the participants were asked a series of questions regarding their thoughts on each tried method, comparing the methods and identifying which one they preferred and which one they disliked. They were further asked about the ramifications of accessibility testing, and how they thought accessibility testing could be integrated into their routines.

Table 1. Data collection

Data	Explanation	Number
Full work days observed	We observed the project members in their daily work	18 days
Stand-up meetings	We observed teams having daily stand-up meetings	15 meetings
Team meetings	We observed meetings in the agile teams, including 2 retrospective meetings	5 meetings
Interviews outside of tests	We interviewed one team leader, one developer, one UX designer, and one architect	4 interviews
USE questionnaires	Seven of Cambridge simulation glasses, seven of SiteImprove, five of dyslexia simulator	19 answers
Testing sessions with end interviews	We tested and interviewed four developers, one software tester, and two UX designers	7 sessions

4.1 The Accessibility Testing Tools

We tested three different accessibility testing tools. Five participants tested three tools while two participants tested two tools because of less time available. The participants were encouraged to think aloud. When they forgot to talk while they were using the tools, we reminded them to do so. Below, we will describe how we tested each of the tools.

SiteImprove. Before we began, the participants were instructed to install the extension on their computers, and they were also given a quick overview of what it does at the start of the session. The SiteImprove user test was conducted without any specific tasks or scenarios in mind, as the tool does not need any extraordinary web features to be able to highlight its capabilities. The participants would explore the tool and investigate the different layers of information on a given website. When they had questions about the tool or were stuck, we guided them in the right direction. After exploring the tool on the given website, five of the participants switched to sites they were working on or were more familiar with to get a better sense of what the tool was telling them.

Simulation Glasses. After a brief introduction and determining how many glasses each tester should wear, the participants who tested the simulation glasses where given tasks to complete while wearing the glasses (Fig. 1). The tasks involved interacting with a web interface and completing two scenarios. The scenarios involved finding and booking airplane tickets online from one of the leading airlines in Scandinavia.

Fig. 1. A participant testing with simulation glasses

Dyslexia Simulation. Before we began, the participants were instructed to install the extension. We described how the tool worked and how it could help them discover accessibility vulnerabilities. Participants were asked to navigate a website with the dyslexia extension turned on and complete two scenarios. The scenarios involved finding information about prices, luggage, and legal requirements. The chosen web pages contained a lot of text to afford the extension something to work with.

4.2 USE Questionnaire for Tool Evaluation

Every participant filled out a questionnaire for each of the methods they tested. The questionnaire for evaluation of the testing methods is a post-session rating metric named the “Usefulness, Satisfaction, and Ease of use (USE) Questionnaire” [25]. The USE Questionnaire is a validated questionnaire that consists of 30 questions

divided into four categories. We considered this questionnaire to be helpful in assessing whether each of the methods was suitable for agile projects, by stating general questions about satisfaction with usability and ease of use on a seven-point Likert scale.

4.3 End Interview After Test

At the end of each testing session, we asked the participants a set of prepared questions to gauge what they thought about the testing tools. They were asked to compare the tools in different ways and to talk about how they experienced using each of the tools. We also asked them whether they believed they would use the tools in the next three weeks, which tool they felt discovered most faults, and how the tools could be integrated into the daily agile workflow routines. Also, we had some general questions about how accessibility aspects affected the products they were developing.

5 Results

From the interviews, it became evident that many project members did not know much about accessibility testing before they tried the tools. One developer suggested that a reason was that they did not learn it in their software engineering education, and he also claimed it was not a common theme in work discussions. From our observations, we could confirm that there was little talking about accessibility among the project members. In none of the 20 observed meetings was there any discussion of accessibility issues. However, all interviewees were positive to learn about tools to help them test for accessibility throughout the agile project. Even though many had little experience with the tools, all participants managed to install the tools in a short amount of time.

5.1 SiteImprove Accessibility Checker

Seven participants evaluated the SiteImprove tool. As Fig. 2 shows, the method did very well, with a high score for all categories. The percentage reflects how well a category scored on the Likert scale from one to seven, where seven is 100% and one is 0%. The tool received a total score of 4.98 ($\sigma = 0.40$) with a 95% confidence interval between 4.68 and 5.28.

SiteImprove scored very high in the Usefulness category with a score of 5.43 ($\sigma = 0.42$), which is well above the average of 4.0. Ease of Learning received a score of 4.82 ($\sigma = 0.41$) since most of the operations are provided automatically by starting the extension.

The method also scored high in Ease of Use with a score of 4.87 ($\sigma = 0.42$) even if some had minor complaints and somewhat struggled: "*The tool points out that there are faults, but it is hard to spot where the faults are*". However, many stated that SiteImprove was much better to use than WCAG. One developer said, *"This is close to a developer tool (…). I like that it makes references to WCAG. I have tried to look at the WCAG documentation, but found that difficult"*.

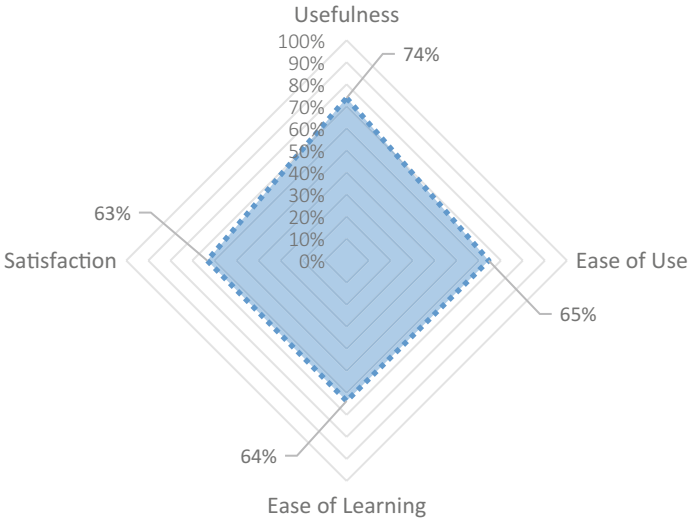

Fig. 2. SiteImprove evaluation results

5.2 Cambridge Simulation Glasses

Seven participants also evaluated this method. As Fig. 3 shows, the glasses got high scores in all categories and had the highest total score of 6.15 (σ = 0.48) among all the methods. The 95% confidence interval for the method was between 5.79 and 6.51.

Not surprisingly, since this method requires the participant only to wear glasses, this method scored very well on both Ease of learning with a score of 6.79 (σ = 0.12)

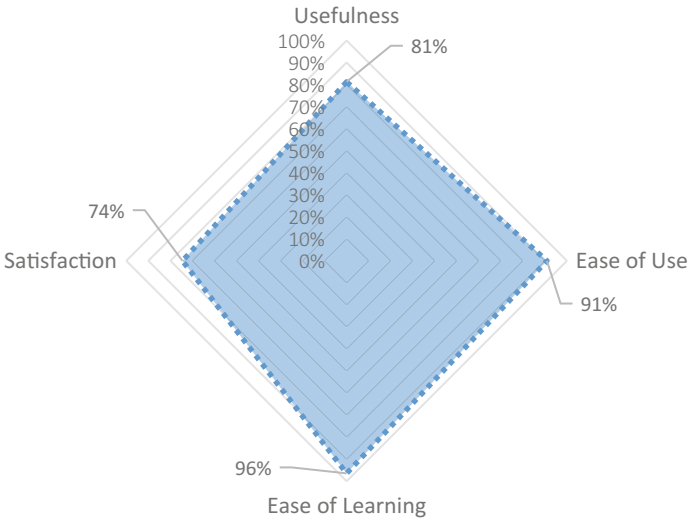

Fig. 3. Cambridge simulation glasses evaluation results

and Ease of use with a score of 6.45 (σ = 0.49), One developer said, "*The glasses gave quick results and were easy to use. I also like that they are tangible*", while another developer said, "*I want to have these glasses on my desk and use them often*".

Also for the Usefulness category, the method scored high with 5.88 (σ = 0.48), and the sub-question "Is it useful" scored 6.57 (σ = 0.73), which is well above the average for the category. A designer said, "*I will use these glasses in meetings between developers and business developers to make them aware of how the different solutions they are discussing will be perceived by people with reduced sight*".

Regarding Satisfaction, the method scored the highest of all with 5.47 (σ = 0.84), and this was also reflected in the interviews. Many participants mentioned that the method was fun to use, and many also tried the glasses on their own after the session was completed. During the interviews, many participants mentioned that the method created more awareness of the challenges associated with bad contrast and small fonts.

5.3 Dyslexia Simulation

Five participants evaluated the dyslexia simulation tool, and as Fig. 4 shows, this method was also regarded highly. The tool had the second highest evaluation in the Ease of learning category with a score of 6.35 (σ = 0.3). This is not unexpected since the method requires only the push of a button in the browser to be enabled. Both Usefulness and Ease of use also had high scores with 5.33 (σ = 0.57) and 5.62 (σ = 0.67). A designer said, "*The dyslexia simulator makes me realize how important it is to use common and simple words because those are easier to recognize when they are scrambled. I will definitely use this tool to show the management how it will be perceived by people with dyslexia if we use long text with difficult words".*

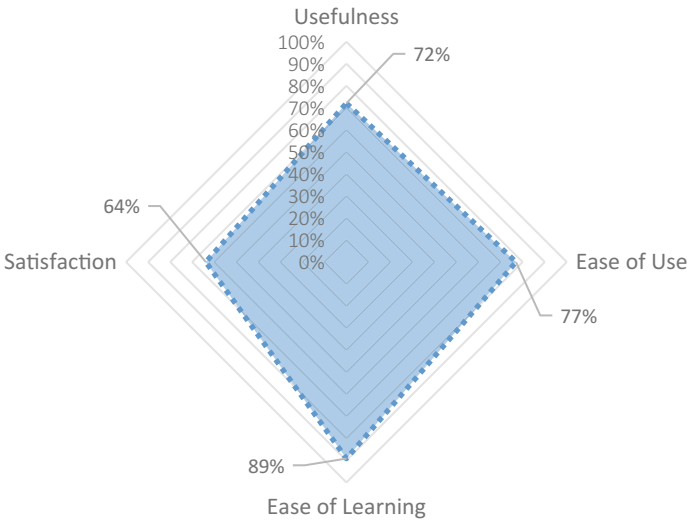

Fig. 4. Dyslexia simulation evaluation results

Overall the Dyslexia simulation method did good, in particular in the category Ease of learning, as Fig. 4 shows. The method itself scored well above neutral (4) and might have received even better scores if not for some bugs in the extension. This is also reflected in the sub-question for Ease of use where the question "I don't notice any inconsistencies as I use it" scored 4.00 ($\sigma = 0.89$), which is well below all the other sub-questions. This is also probably connected to the fact that the extension did not adapt all the text on the webpage.

The tool scored well overall with a total score of 5.53 ($\sigma = 0.53$) and a 95% confidence interval between 5.14 and 5.92. In the interviews, many participants said that the method was an eye-opener experience. Participants also liked that it was easy to visualize problems with too much text on a web page and that it emphasizes the importance of writing good and readable text: One developer said, *"I like this plugin. I now see that the way you structure the text is very important. When you have too much text close together, it is hopeless for a person with dyslexia"*. Contrarily, some noted that it could be difficult to understand how good the method was at finding issues.

6 Discussion

Good accessibility is difficult to achieve, mainly due to its complex nature and the wide range of people who must be considered. It does not help that accessibility testing is conducted late in the development process when changes are costly to make, and compromises are made instead. However, agile development can significantly help to improve web accessibility due to its focus on regular testing [9]. Eklund and Levingston [6] and Ferreira et al. [10] argue for more frequent testing where more cost-effective methods are used several times throughout the project rather than relying on few but large and costly testing activities, such as hiring accessibility experts to do accessibility testing.

There is currently little research that offers concrete solutions for simple accessibility testing in agile settings; most current research discusses just the possibilities for accessibility testing in agile development. In this study, we have focused on giving accessibility testing some much-needed evaluation in an agile project. The fast and low-cost test methods in this study were perceived as useful and easy to use by the agile project members.

Next, as other research also has suggested [9, 26], we found that accessibility needs to be a team effort. Agile principles promote self-organization and communication within the development team and make it possible to inspect even the smallest of details when everyone is involved, instead of having only a few experts working on accessibility. Several of the methods we tested addressed many of these concerns. While some were easier to use and learn than others, they all were usable without extensive training and learning sessions. Our results showed that all three methods we tested, SiteImprove, the Cambridge glasses, and the dyslexia simulator, can be used across the development team and utilized in such a way that it becomes a team effort where testing can be commenced at regular but short intervals.

All three accessibility testing methods we used in this study were low-cost and perceived as very easy and quick to use. Most of them can be used very early in development, some even before any code is written, merely going by design drawings. There is no doubt that accessibility testing can be included early in agile software projects. Using any of the tools will enable testing to be done more often. The interviewees also suggested regular workshops at which the agile team members together could experience the solutions they made with different accessibility testing tools. Also, if there are formal requirements from the organization or customer, these issues could be discussed in planning and retrospective meetings and would probably be a more frequent theme in the daily-standup meetings.

SiteImprove was quick in producing results and is the method that would cover the broadest set of WCAG paragraphs in our study. One key factor in its success is the automation aspect of the tool. It dramatically reduces the time spent finding bugs, and the user is immediately presented with clearly defined statements of what is wrong and how one can improve. It was not surprising that the participants rated SiteImprove as easy to learn since most of the operations are provided automatically by starting the extension. We had expected a lower total score of SiteImprove since the tool uses many complicated terms and advanced terminology. A drawback with SiteImprove is that it cannot be used until a solution has been implemented, rendering it not as useful for designers or in the early phases. It is also limited by being able only to check what is programmatically possible.

The Cambridge simulation glasses were, along with SiteImprove, the method that worked well in many situations. It also worked well in cases where SiteImprove cannot help. Use of the glasses is not dependent on a solution having reached a certain level of development. The glasses can be used early in development on visual concepts, for example on sketches or prototypes. They are also platform-independent and can be used on any software or visual representation, making it a universal accessibility tool. One aspect remarked on by many of the participants who tested the method was that the glasses gave an overview of the product in its entirety. Being able to view the product from a higher perspective rather than fixating on small details also contributes to why so many liked to use the glasses.

We included the dyslexia tool because we wanted methods that covered cognitive impairment, and this method was one of the few testing methods that are low-cost and fast to test for this. The method was rated higher than we had foreseen, particularly because we expected more participants to misunderstand the idea behind the method. The tool received good results on the USE Questionnaire. However, with this tool, the testers have to interpret the results based on their knowledge of dyslexia. Some testers reported that the plugin could be helpful to discover text where long and complicated words were being used, and where the structure and layout were not organized. The improved accessibility by employing clear and concise language is an aspect not addressed by any of the other tools.

6.1 Threats to Validity

As with all empirical studies, there are some threats to validity that we need to consider. First, we used a single-case design and, as a result, have the possibility of bias in the

data collection and analysis. Therefore, the general criticisms of single-case studies, such as uniqueness, may apply to our study. However, as the company had been using agile methods for many years, developing bank solutions for a wide variety of people, we found it to be a valuable case for investigating accessibility testing in agile software development. Furthermore, we triangulated the research question from multiple sources of evidence such as interviews, observations, and questionnaires to increase the quality of the case study [12].

Second, when using self-reported data to measure usability of tools, one might have the "social desirability bias" [27] where respondents respond more positively to rating questions because they unconsciously want to satisfy the facilitator giving them the questionnaire. To reduce this bias, following the advice of Albert and Tullis [25], we collected the responses anonymously on the web, and we did not look at the results until afterward.

6.2 Implications for Practice

Our results show that agile accessibility testing methods are not well-known among agile team members. Accordingly, agile projects should invest in simple tools that can be used by all roles, and project members should be informed about the techniques they can use. We found that the three tools tested were useful and should be used regularly in agile projects as they are also cost-efficient. For software developers who consider using accessibility testing tools, we suggest that Cambridge glasses be used early, as they do not require much effort from the agile developers. As the product becomes more stable, SiteImprove can be used regularly.

The methods and tools we have evaluated have been used with the intention of testing software and finding accessibility issues. In the larger scheme of accessible software, we suspect that using these tools, especially the ones that simulate an impairment, might have secondary benefits. Exposure to the type of simulation testing such as with the Cambridge glasses and the dyslexia simulator can give the tester a heightened sense of awareness of issues that could come into conflict with impaired users and foster more empathy and understanding.

Many comments were made that the testers had new insights and understanding. The new knowledge about how different users perceive software can give developers a subconscious ability to shape software in a more accessible direction, outside of directly applying accessibility methods. So, while the methods described in this paper might not be chosen to be used regularly, there can be a significantly insightful experience for agile developers to try these methods, and they can put that experience to use in later development.

There is still a way to go to more easily integrate accessibility testing into agile work processes. The existing tools should be more compatible with agile processes, and agile project members should have a wider variety of accessibility testing tools to aid them in making accessible software. Implementing this would reduce the resources needed for accessibility testing by continuing the work toward more lightweight and comprehensible tools, as well as making the surrounding issues with impairments more visible and the knowledge of how they affect software interaction more obtainable. This will aid in weaving accessibility into the development process.

Another barrier is that many testing tools are platform-dependent. Tools are not uniformly available on all operating systems, browsers, and developer tools. The different tools have their own characteristics, and integration and learning can be made more arduous because of this. Tools that support software not intended for web or mobile are also rare. The lack of diverse tools available on many platforms hampers any attempt to make accessibility a more uniform process as it now has to be individually tailor-made depending on the hardware and software available to developers, and without the right combination, it can be challenging to cover a wide range of impairments with tools.

7 Conclusion and Further Work

We have evaluated three different tools for accessibility testing, in addition to interviewing the participants and observing them in their daily work in software development teams. We argue that the methods we tested will help to discover accessibility issues while keeping costs low in an agile project. In our study, when introduced to the tools, the project members became more aware of how to make the software more accessible, and they had a positive attitude toward using the tools throughout their projects. The interviewees stated that they wanted accessibility work to be a team effort, rather than the responsibility of the UX designers or accessibility consultants.

Further work should test other accessibility tools such as screen reader, WCAG, and Personas to try to find an approach for these tools to be integrated into an agile development project. Furthermore, future work should investigate the connection between the cost of doing accessibility work throughout the agile process with the cost saved by fixing accessibility issues early.

References

1. Petrie, H., Bevan, N.: The evaluation of accessibility, usability, and user experience. In: Stephanidis, C. (ed.) The Universal Access Handbook, pp. 1–16. CRC Press (2009)
2. Sánchez-Gordón, M.-L., Moreno, L.: Toward an integration of Web accessibility into testing processes. Procedia Comput. Sci. **27**, 281–291 (2014)
3. Kane, D.: Finding a place for discount usability engineering in agile development: throwing down the gauntlet. In: Proceedings of the Agile Development Conference, pp. 40–46. IEEE (2003)
4. Zimmermann, G., Vanderheiden, G.: Accessible design and testing in the application development process: considerations for an integrated approach. Univ. Access Inf. Soc. **7**, 117–128 (2008)
5. Nielsen, J.: Return on Investment for Usability. Jakob Nielsen's Alertbox (2003)
6. Eklund, J., Levingston, C.: Usability in Agile Development, pp. 1–7. UX Research (2008)
7. Stray, V., Moe, N.B., Hoda, R.: Autonomous agile teams: challenges and future directions for research. In: Proceedings of the XP2018 Scientific Workshops, Porto, Portugal. ACM (2018)
8. Mikalsen, M., Moe, N.B., Stray, V., Nyrud, H.: Agile digital transformation: a case study of interdependencies. In: Proceedings of the Thirty Ninth International Conference on Information Systems (ICIS 2018), San Francisco (2018)

9. Luján-Mora, S., Masri, F.: Integration of web accessibility into agile methods. In: Proceedings of the 14th International Conference on Enterprise Information Systems (ICEIS 2012), pp. 123–127 (2012)
10. Ferreira, J., Noble, J., Biddle, R.: Agile development iterations and UI design. In: Agile Conference (AGILE), pp. 50–58. IEEE (2007)
11. Fuglerud, K.S.: Inclusive design of ICT: The challenge of diversity (2014)
12. Yin, R.K.: Case Study Research: Design and Methods. Sage, Thousand Oaks (2009)
13. Bias, R.G., Mayhew, D.J.: Cost-Justifying Usability: An Update for the Internet Age. Morgan Kaufmann Publishers Inc., San Francisco (2005)
14. Bai, A., Mork, H.C., Stray, V.: A cost-benefit analysis of accessibility testing in agile software development: results from a multiple case study. Int. J. Adv. Softw. **10**, 1 (2017)
15. Haskins, B., Stecklein, J., Dick, B., Moroney, G., Lovell, R., Dabney, J.: Error cost escalation through the project life cycle. In: INCOSE International Symposium, pp. 1723–1737. Wiley Online Library (2004)
16. Bordin, S., De Angeli, A.: Focal points for a more user-centred agile development. In: Sharp, H., Hall, T. (eds.) XP 2016. LNBIP, vol. 251, pp. 3–15. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-33515-5_1
17. Chamberlain, S., Sharp, H., Maiden, N.: Towards a framework for integrating agile development and user-centred design. In: Abrahamsson, P., Marchesi, M., Succi, G. (eds.) XP 2006. LNCS, vol. 4044, pp. 143–153. Springer, Heidelberg (2006). https://doi.org/10.1007/11774129_15
18. Moreno, A.M., Yagüe, A.: Agile user stories enriched with usability. In: Wohlin, C. (ed.) XP 2012. LNBIP, vol. 111, pp. 168–176. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30350-0_12
19. Meszaros, G., Aston, J.: Adding usability testing to an agile project. In: Agile Conference, pp. 289–294. IEEE (2006)
20. Bai, A., Fuglerud, K., Skjerve, R.A., Halbach, T.: Categorization and comparison of accessibility testing methods for software development. Stud. Health Technol. Inf. **256**, 821–831 (2018)
21. W3C Working Group Reading Level: Understanding Success Criterion 3.1.5. https://www.w3.org/tr/understanding-WCAG20/meaning-supplements.html
22. Stray, V., Moe, N.B., Sjøberg, D.I.K.: The daily stand-up meeting: start breaking the rules. IEEE Softw. (2018). https://doi.org/10.1109/ms.2018.2875988
23. Derby, E., Larsen, D., Schwaber, K.: Agile Retrospectives: Making Good Teams Great. Pragmatic Bookshelf (2006)
24. Edwards, R., Holland, J.: What is Qualitative Interviewing? A&C Black (2013)
25. Albert, W., Tullis, T.: Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics. Newnes (2013)
26. Constantine, L.: What do users want? Engineering usability into software. Windows Tech J. **4**, 30–39 (1995)
27. Nancarrow, C., Brace, I.: Saying the "right thing": coping with social desirability bias in marketing research. Bristol Bus. Sch. Teach. Res. Rev. **3**, 1–11 (2000)