A Cost-Benefit Analysis of Accessibility Testing in Agile Software Development

Results from a Multiple Case Study

Aleksander Bai
Norwegian Computing Center,
Oslo, Norway
Email: aleksander.bai@nr.no

Heidi Camilla Mork
Kantega AS,
Oslo, Norway
Email: heidi.mork@kantega.no

Viktoria Stray
University of Oslo,
Oslo, Norway
Email: stray@ifi.uio.no

***Abstract*—It is important to include accessibility testing in software development to ensure that the software developed is usable by as many people as possible, independent of their capabilities. Few guidelines exist on how to integrate accessibility testing in an agile process, and how to select testing methods from a cost-benefit point of view. The end result is that many development teams do not include accessibility testing, since they do not know how to prioritize the different testing methods within a tight budget. In this paper, we present an evaluation of nine accessibility testing methods that fits in an agile software development process. We discuss the cost of each method with regards to resources and knowledge requirements, and based on a cost-benefit analysis, we present an optimal combination of these methods in terms of cost and issues discovered. Finally, we describe how accessibility testing methods can be incorporated into an agile process by using the agile accessibility spiral.**

Keywords*—*Accessibility testing; Agile software development; Cost-benefit analysis; Usability;

I. Introduction

Accessibility testing in software development is testing the software to ensure that it is usable by as many people as possibly, independent of their capabilities. The past decade have seen an increased interest in integrating usability in the software development process. However, far to little attention has been paid to the field of accessibility, and how to incorporate accessibility testing in software development. Earlier we have described a cost-benefit approach for selecting accessibility testing methods [1]. In this paper, we further investigate and develop this approach with an additional case study.

There is an increased focus on accessibility from legislation and the United Nations with "Convention on the Rights of Persons with Disabilities" [2]. While usability focus on the quality and ease of use for a product, accessibility focus on letting people with the widest range of capabilities be able to use a product [3]. However, both definitions are overlapping and within the spirit of universal design: the design of products and services to be usable by all people, to the greatest extent possible [4].

Studies show that doing usability testing is costly and can take around 8-13% of the project's total budget [5]. Much of the cost goes to recruiting participants and evaluators in addition to the man-hours required for conducting and evaluating the results [6]. For accessibility testing, the cost can be even higher than usability testing, since recruitment and accommodation of participants usually have more requirements.

However, by not doing accessibility testing at all or by postponing testing until the end of the project, the cost can be extremely high, and it might not even be possible to do accessibility adjustments at a late stage [7] [8]. Many studies show that software that is hard to use, or have features that are hard to understand, make users find better alternatives [9]. There might also be legal requirements to provide accessibility, like in the European Union and in the US.

Our approach is targeted towards agile software development, since it has become the mainstream development methodology [10]. According to a recent survey [11], more than 70% of developers work in agile software projects. Agile software development emphasise delivery of working software in short development iterations [12], active customer engagement [10], frequent communication through daily stand-up meetings [13] and integration of development and testing [14]. All these principles make it feasible to integrate accessibility testing in the whole development process.

We argue that developers, testers and designers in agile software teams can take more responsibility for accessibility testing, and thus lower the total testing cost of the project and at the same time deliver a better product that is both more usable and accessible. Jurca et al. [15] found that one of the major problems with integration of Agile methods and User-centered design (UCD) was that interaction designers are often overworked and distributed across several projects. Thus, it is beneficial that developers and testers in agile software teams also take responsibility for accessibility testing.

During our evaluations, we have investigated different accessibility testing techniques, and we discuss the cost-benefit aspect of these in an agile development process. We argue that accessibility testing does not necessarily require a high cost. We describe where in the process the methods can be used and how they can be combined in optimal ways. Consequently, the impact from accessibility testing towards the end of the project will be minimized, and thus reduce the cost of retrofitting [7]. Our suggested approach is not a substitute for doing user testing, but an addition, incorporated into the agile development process, to reduce the overall cost and increase the usability and accessibility of the software.

The remainder of this paper is organized as follows. Section II summarizes related work, and Section III gives an overview of accessibility testing methods. Section IV describes the evaluation setup and the two case studies, and the results from

the studies are presented in Section V. Section VI reports our cost-benefit analysis of the accessibility testing methods and Section VII discusses the results and implications. We have presented limitations in Section VIII, before we summarize and conclude in Section IX.

II. Related Work

Zimmermann and Vanderheiden [16] have proposed a method for integrating accessible design and testing in the development of software applications, both iterative and non-iterative processes. However, the proposed method's main focus is on how to gather accessibility requirements and does not contain much details on how to actual perform testing in an iterative process. There has been some focus on integrating an agile development process with usability testing [17], and in recent years, there has been an increasing interest in Agile UX (User Experience) [18]. Bonacin et al. [19] propose how to incorporate accessibility and usability testing into an agile process, but do not discuss which accessibility testing methods that are optimal to use or how to combine them in an efficient setup. Horton and Sloan [20] have proposed a accessible user experience framework for organizations on how to integrating accessibility in the design and development process.

There has been research on how to conduct a cost-benefit analysis of universal design of ICT [21], but there is a lack of case studies to verify the concepts. The research focus has also primarily been on the overall effect of universal design, and not the testing methods.

A recent systematic review of usability testing methods for software development processes [9] requests more research into evaluating the different testing methods and how they affect the testing outcome. To the best of our knowledge, there are no evaluations of accessibility testing methods in an agile process, and there are no studies of which accessibility testing methods that are most effective compared to resources and knowledge available in a agile team. We address the latter issue in this paper by showing a cost-benefit approach on how to select accessibility testing methods in an agile process.

Most agile team members have good knowledge and focus on both testing and usability, but it is much rarer to consider accessibility as part of the development cycle. This is probably caused by little knowledge and experience with accessibility testing, cost and time constraints associated with accessibility testing, and available resources to perform accessibility testing. According to [22], who surveyed participants involved in web development projects, only 20% consider accessibility aspects in their projects and nearly half said that they do not use any accessibility evaluation methods.

III. Background

Nowadays there is a widespread application of agile methods in software projects. Agile software development is a set of principles for developing software iteratively, in order to react rapidly and flexible to changing requirements [12]. Agile development teams have given more attention to the importance of usability the last decade, and interaction designers are often part of the agile teams. User-centered design (UCD) is a design process where the needs, requirements and capabilities of the end-users are considered throughout the entire design- and development life cycle. The integration of agile principles with those from UCD is called Agile-UX. In recent years, there has been an increasing interest in Agile UX [18], but little attention has been paid to the field of accessibility.

Accessibility is often divided into two parts; *technical accessibility* and *usable accessibility* [23]. Technical accessibility is to follow good technical standards when developing software, like giving alternate text to images or use the correct heading tag. Usable accessibility is to ensure that the solution is usable to the widest range of users as possible in the sense that users can understand, navigate and interact with the solution. Software can be technical accessible but still not usable if the user does not understand how to interact with the system in order to solve a task. Note that there is rarely a clean separation between technical and usable accessibility and the issues are often overlapping.

There are several methods for testing usability [9] and accessibility [16] in software development: automated checkers, guidelines, expert walkthrough, interviews and user testing, to name a few. The different testing methods uncover different kinds of issues. Automated checkers cover technical accessibility, whereas methods closer to user testing cover more of the usable accessibility.

Automated checkers are tools that a developer or tester can run to get an evaluation of issues regarding the technical accessibility. There are numerous alternatives out there [24], like the NetBeans accessibility module [25] that integrate directly into the developer's tools, or tools that can be used as a step in continuous integration. However, automated checkers only check issues that can be detected programmatically, i.e., issues related to technical accessibility.

There are many different tools, kits or wearables that a person can use to simulation various types of disabilities. The motivation is to let a person experience an impairment so the person might be able to gain some insight into the issues that an impairment might have with a certain design or solution [26]. It is important to note that the intention of a simulation kit is not to simulate the disability in itself, which is highly criticized [27]. Simulation with wearables, also refered to as screening techniques [28], will uncover issues mainly in the area of usable accessibility, but it will also find issues with technical accessibility, for instance low color contrast between text and background.

Testing with assistive technology like a screen reader or computers' high contrast mode will make issues with the technical accessibility very obvious as assistive technology is highly dependant of proper coding to function properly.

Checklists and guidelines provide the evaluator with a set of instructions and criteria to evaluate, and the WCAG (Web Content Accessibility Guidelines) 2.0 standard [29] is the de facto choice. Checklists cover both technical and usable accessibility.

Persona walkthrough or persona testing approach [30],

is where an expert simulates or play-acts a persona while carrying out tasks. The more knowledge the expert has about the disability that a particular persona has, the easier it is to do a realistic and credible acting while testing the solution. A persona walkthrough will usually cover usable accessibility.

There are many approaches on how to group accessibility testing methods [31], and we have chosen to group the testing methods into five groups as shown in Table I. The table shows different groups of testing methods with an indication of whether the group mainly discovers technical or usable accessibility issues, or issues of both types.

TABLE I. ACCESSIBILITY TESTING GROUPS.

#	*Group*	*Technical accessibility*	*Usable accessibility*
1	Automated checkers	x	
2	Checklist and guidelines	x	x
3	Simulation kits		x
4	Assistive technology	x	
5	Walkthrough		x

The best approach for accessibility testing is user evaluation, since the actual users are involved and the testers does not have to do any approximation of impairments or mental states [32] [33]. However, it is also an expensive method because it requires much planning, recruitment and management [6]. It is also particular important to find as many problems as possible before involving users. Examples of user evaluation involve user testing, interview and focus groups. The focus of our study was to investigate testing methods that members of an agile team can use themselves during the various phases of the development process. User testing is therefore outside the scope of this paper.

IV. CASE STUDIES

To conduct a cost-benefit study, we at least need to consider resources requirements of the methods, including knowledge requirements. We also need to consider what kind of issues the different accessibility testing methods can discover, and how the test methods differ from each other. A further goal is to investigate where in an agile development process the different accessibility test methods might be most valuable.

We conducted two case studies with two different software solutions. After the first case study we performed a cost-benefit analysis, and the result of this analysis was used to suggest which methods to use and when to apply the different methods in an agile development process. In the second case study we wanted to further investigate the most valuable testing methods found in the cost-benefit analysis from the first case.

In each case we did a selection of accessibility testing methods, where each selected method was tested by at least two persons, and the results were aggregated. One person performing one test method is called an *evaluation*. Each evaluation had a coordinator who wrote down the issues reported by the tester, and the coordinator also made notes when difficulties, that were not verbally expressed, were observed. The registered issues were then classified as critical, cognitive, both or none of them. We defined a *critical issue* as an issue that prevents the participant from continuing or completing a task; e.g., difficult to read images or text because of poor contrast or resolution. A *cognitive issue* was an issue caused by confusing or missing information for the given context; e.g., not understanding the purpose of a screen or not understanding how to operate a controller. A problem can thus belong to both the critical and cognitive category, as it was often the case. The number of issues for each of the testing methods, together with the number or critical and cognitive issues, formed the basis for further analysis.

For the case studies, we selected methods from all the groups in Table I. From Group 1 we used the WAVE Web Accessibility Evaluation Tool [34], which is an online tool that evaluates the website of any given available URL. The user provides the URL and the tool gives a report on the errors on the web page. We selected the WCAG (Web Content Accessibility Guidelines) 2.0 standard [29] and the VATLab checklist [35] from Group 2. From Group 3 we used Cambridge inclusive design glasses [36] for simulating reduced vision, and the Cambridge inclusive design gloves [36] to simulate dexterity reduction. The gloves are typically used for testing a physical product, but they were included in the first case since there was a card reader involved. From Group 4, we used the screen reader NVDA [37] and the built-in high contrast mode in Windows. We used dyslexia and old male for persona walkthrough from Group 5.

In the first case study we had two participants with the necessary knowledge and experience to do persona walkthrough with one persona as old male and one with dyslexia.

The total set of methods used in the case studies is shown in Table II.

TABLE II. SELECTED METHODS FOR CASE STUDIES.

	Method	*Case A*	*Case B*
M1	Automated checker		x
M2	Simulation kit reduced vision	x	x
M3	Simulation kit reduced dexterity	x	
M4	Assistive technology screen reader	x	x
M5	Assistive technology high contrast	x	x
M6	Manual WCAG	x	x
M7	VATLab checklist	x	x
M8	Persona dyslexia	x	
M9	Persona old male	x	

The different testing methods vary in how much resources and knowledge it takes to use them. In each case study we categorized the methods as either *low*, *medium* or *high* in required amount of resources and knowledge. In terms of resources, *low* means that none or little prerequisites (tools, setup, administration) are required to conduct the method; *medium* means that some prerequisites are required, and they are relatively cheap (under $1000); *high* means that the method requires considerable investments in terms of setup, purchase, administration or maintenance. In terms of knowledge, *low* means that no or very little prior knowledge is required; *medium* require some prior knowledge, either technical (usage, commands) or domain (knowledge or experience with the impairment); *high* means that extensive or expert training is

required to conduct the evaluation. These categories will be used for the cost-benefit analysis in Section VI.

A. Case A: e-ID

The solution used for evaluation in case A was a pilot for electronic identification, developed during the EU project FutureID [38]. The pilot uses a software certificate or an ID card with a card reader for the authentication process. The pilot uses both a Java client and a web front end, and consists of around ten different user interfaces with varying complexity.

1) Selected Methods: We used eight of the methods from Table II and labelled them with the categories *low*, *medium* and *high* with regard to the requirements of resources and knowledge.

Almost all methods are labelled as *low* with regard to resources. The only methods with prerequisites were the simulation kits because some hardware must be purchased ahead of testing. This is usually a one-time purchase, but it must also be stored and assembled before usage, so we think it qualifies as medium resource demanding compared to the others.

Most of the methods require very little prior knowledge. Method M4 involves using a screen reader, which requires knowledge on how to operate it, but almost everyone can learn how to use a screen reader in a reasonable amount of time, and therefore we labelled it *medium*. However, our level of using screen readers cannot compare with the expert level of people that use screen readers daily and are dependent on them.

The persona walkthrough is informal and relatively quick to do but is heavily dependent on the selected personas and the experience that the expert has with the particular type of disability. The persona walkthrough methods require expert knowledge, and is thus marked with *high* knowledge requirements. However, there are few resource requirements for persona walkthrough methods, and this is why they are labeled with *low* for resource requirements.

Table III gives the summary of the selected methods, and more details can be found in [1].

TABLE III. OVERVIEW THE SELECTED METHODS

Method	*Resources requirements*	*Knowledge requirements*
M2	Medium	Low
M3	Medium	Low
M4	Low	Medium
M5	Low	Low
M6	Low	Low
M7	Low	Low
M8	Low	High
M9	Low	High

2) Participants: Five different participants performed the evaluations, where the participants' knowledge on accessibility testing ranged from beginner to expert. All the participants had technological background, their age ranged from 35 to 61, and there were both males and females in the group. Two of the participants where recruited based on their experience with persona testing and their knowledge on dyslexia and aging. The participants performed a total of 17 evaluations.

TABLE IV. PARTICIPANTS P1 – P5 AND THE METHODS THEY EVALUATED

Method	P1	P2	P3	P4	P5
M2	x	x	x		
M3	x	x			
M4	x	x			
M5	x	x			
M6	x	x			
M7	x	x			
M8				x	x
M9				x	x

3) Procedures: All the evaluations were conducted on the same machine with the same setup to ensure an equal test environment. A short initial test was conducted before the evaluations started, in order to verify that the overall setup, the scenarios, and the ordering of them were best possible. The goal of all the scenarios was to successfully log into the solution. Each participant performed five different scenarios in the same order:

1) Invalid digital certificate
2) Valid digital certificate
3) Invalid smart card
4) Valid smart card, but incorrect PIN code
5) Valid smart card and correct PIN code

The participants were unaware that they were given invalid certificate, invalid smart card and invalid PIN (Personal Identification Number) code. The scenarios were also executed in the listed order to avoid biasing the participants as they should gradually progress a little further in the login process. Each method took under two hours to complete for a single participant for all the scenarios.

B. Case B: pension

The solution used for evaluation in Case B is a Norwegian public available website, Norsk Pensjon [39], used by citizens to get an overview of their personal pension schemes. The key functionality of the website is a calculator where one can estimate future pension payments based on parameters like current salary and the expected year of retirement. The company behind the website is concerned with universal design and accessibility. The website has recently been evaluated by accessibility experts, and there is a plan to improve the site accordingly. The website is developed by the IT consultancy company Kantega, and current members of the development team were participants in the evaluation. The focus of this second case was to further verify the results of the first Case A and see how the testing methods worked for participants in agile teams. An important aspect with Case B was to evaluate the key findings from case A in real environments with actual developers and testers.

1) Selected Methods: The methods for this second case was chosen based on the cost-benefit analysis of the first case and the agile accessibility spiral [1] that fits in an agile development process. We also chose to include an automated checker that were not included in Case A, since automated tools in general require fewer resources than other methods. We hypothesised that an automated checker could be a natural

first choice for any agile team, even though they only discover issues that are detectable programmatically. The selected tool, WAVE [34] is free to use, and thus *low* in required resources. WAWE gives reports with explanations to the operator, hence very little prior knowledge is required and we labelled it *low*.

We used the Cambridge inclusive design glasses [36] to simulate reduced vision, but we did not test with the simulation kit for reduced dexterity since no extra device, like a card reader, was involved. The testing methods assistive technologies screen reader and high contrast were also selected. All these methods were labelled with the same knowledge and resource requirements as in Case A. We also included the manual WCAG evaluation and the VATLab chekclist, based on how they scored in Case A. However, we experienced that the checklists were more difficult to use than expected, and we therefore decided to label them as medium in knowledge requirements in Case B. We chose to not include persona testing in this second case since none of the participants or teams had experience with this type of testing.

TABLE V. OVERVIEW OF THE SIX METHODS

Method	*Resource requirements*	*Knowledge requirements*
M1	Low	Low
M2	Medium	Low
M4	Low	Medium
M5	Low	Low
M6	Low	Medium
M7	Low	Medium

2) Participants: There were in total six participants. The participants were a designer and a developer of the current development team for the website, along with other developers and technologists. Their age ranged from 25 to 37, and there were both males and females in the group. The participants performed a total of 22 evaluations.

TABLE VI. PARTICIPANTS P1 – P6 AND THE METHODS THEY EVALUATED

Method	P1	P2	P3	P4	P5	P6
M1	x	x	x	x		
M2	x	x	x	x		
M4	x	x	x	x		
M5			x	x		
M6	x	x	x	x	x	x
M7			x		x	

3) Procedures: The evaluations with methods from the simulation group were conducted with a coordinator, and one or two observers who registered issues. The participants performed five scenarios in the same order:

1) Calculate the pension payment at age 80 if you retire at age 63 and currently have a salary of 500 000 NOK
2) Send the payment plan by email in the case where you retire at age 70 and have a salary of 500 000 NOK
3) Find the estimated payment at age 68 if you take partial retirement at age 66 with 50% work position and you want to take out 40% of the pension. You have a salary of 500 000 NOK and take full retirement from age 70
4) Find the payment at age 70 given in percentage of your current salary, when you retire at age 63 and have a salary of 500 000 NOK
5) Find information about contractual pension (AFP)

The scenarios were ordered so that it gradually revealed more of the functionality in the solution. After completing an evaluation, the participant was asked how it was to perform the evaluation and to reflect on the use of the test method.

The selected methods from the checklist group, manual WCAG and VATLab checklist, were conducted individually and remotely by the participants. The participants received a checklist with success criteria to meet, together with instructions to mark a criteria in the checklist as failed if they could find an example which did not meet the criteria, otherwise mark it as passed. If the criteria was evaluated as not applicable or relevant for the solution, the criteria was marked n/a.

V. EVALUATION RESULTS

The evaluation results from the case studies in Section IV are presented here, together with a comparison on how the testing methods performed in the two cases.

A. Results from Case A

As can be seen from Table VII, a high number of critical issues were discovered with most of the methods. It should be noted that a critical issue might only be critical in the context of a given disability. For instance, incorrect HTML tags can be critical for blindness, but may not be relevant for impairments like reduced dexterity. However, for the solution as a whole, all critical issues are equally relevant since an issue might exclude some users if nothing is done to improve the problem.

TABLE VII. ISSUES FOUND IN CASE A.

Method	*Issues*	*Critical*	*Cognitive*
M2	54	14	9
M3	4	0	0
M4	33	26	0
M5	10	4	0
M6	32	7	5
M7	19	17	0
M8	34	15	15
M9	27	13	9
	213	96	38

The simulation kit methods (M2, M3) found a lower percentage (25.9%) of critical issues compared to the other methods. The main reason was that most of the issues reported by these methods were visual problems that were annoying at best, and in some cases problematic, but not marked as critical since it did not hinder further progress. Of the critical issues discovered with simulation kits, almost all were also marked as cognitive. Note that a relative few number of issues were found when simulating reduced dexterity, and we believe this is mainly because the application did not require much motoric precision. This is of course very dependent on the software that is evaluated.

Manual WCAG (M6) also reported relative few critical issues (21.9%), and this was mostly because the WCAG evaluations criteria are high level. For instance did a single criteria in WCAG cause over 17 critical issues to be reported in the screen reader method (M4) since it has a much finer

granularity. The WCAG evaluations also reported few cognitive issues for the same reason.

The screen reader testing method (M4) identified the most critical issues (78.8%), and many of the issues were related to poor compatibility with screen readers. We suspect that more issues could be found if there had not been so many critical problems that made further investigations in many cases impossible. High contrast testing method (M5) did not find many issues, and even fewer critical issues, and this is mainly because contrast was not a big problem in the solution. VATLab checklist (M7) found a very high number of critical issues, and this is because the focus is on compatibility with screen readers, and as already explained the support for screen readers was poor.

Persona testing methods (M8, M9) found the most cognitive issues, and this is not unexpected since the persona used in the evaluations focused on usability and understanding the context. Most of the issues reported by persona testing were directly related to the participant not understanding the context of a page and what was expected from the participant. A high number of the cognitive issues were also marked as critical since it is impossible for the participant to complete his task, and this explains the high number of critical issues discovered by persona testing.

B. Results from Case B

The results from Case A guided us in choosing the methods to investigate further in Case B. We chose automated checker, simulation kit reduced vision, screen reader, high contrast and checklists, as can be seen in Table VIII. Also here it must be noted that a critical issue might only be critical in the context of a given disability. This is discussed further in Section VIII.

TABLE VIII. ISSUES FOUND IN CASE B.

Method	*Issues*	*Critical*	*Cognitive*
M1	27	0	0
M2	58	8	17
M4	16	7	5
M5	16	2	5
M6	25	5	6
M7	15	8	0
	157	30	33

All methods found a reasonable amount of issues, but relatively few critical issues, at least compared to what was found in Case A. This was partly expected since the solution used in Case B were in production and less complex than the solution in Case A. However, more cognitive issues were found in Case B. In Case A, only 17.8% of the issues reported was considered cognitive while 21.0% of the issues in Case B were considered cognitive. This is probably because the domain (pension) in Case B was more complex than Case A (login), and thus more cognitive issues were found. It should be noted that few of the cognitive issues were marked as also being critical.

The simulation kit with reduced vision (M2) found considerably more issues compared to the rest, and this is mainly because of visual problems that were annoying, but not critical since it did not prevent progression. This testing method also found most cognitive issues, and this is because reduced vision caused the tester to miss or misunderstand important information.

WCAG (M6) found many issues and managed to identify a large number of cognitive issues in addition to some critical. WCAG in Case B found considerably more cognitive issues than in Case A, and this was partly due to a revaluation of the WCAG checkpoints as explained in Section IV, but also because correct delivery of content and error messages are even more critical in a complex domain such as pension.

The automated checker (M1) found many issues, but none that were considered critical or cognitive. This is because the identified issues were only trivial issues like minor contrast problems and not fundamental problems. It was also a problem that the automated checker could not be used for a solution that uses a secure connection and two-factor authentication, and could thus only be used on the static pages and not the more complex pages. We suspect more issues would be found otherwise, but not necessarily more critical or cognitive.

Both the screen reader (M4) testing method and high contrast (M5) method found a reasonable amount of issues, but of very different types. The screen reader testing method revealed poor support for screen readers, while the high contrast found issues connected to poor contrast mode in essential images in the solution. The VATLab checklist (M7) found many of the same issues as the screen read method.

In Case B, external experts had evaluated the accessibility of the solution. The findings from the experts and the findings from our case study with internal developers and testers showed that we found more or less the same type of issues. This further indicates that doing internal accessibility testing is crucial and worth the investment.

C. Comparison of both cases

Four of the testing methods were the same for Case A and B, and it is interesting to see which of these four methods that identified most issues in total, most critical issues and most cognitive issues. A heat map of the issues found is shown in Figure 1, where green indicate high percentage and red indicate low percentage.

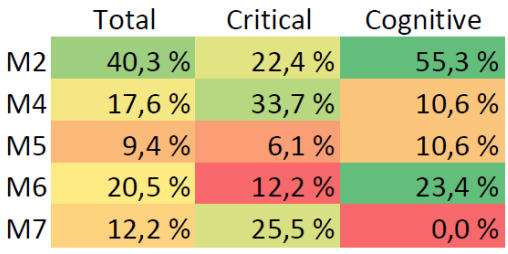

FIGURE 1. HEAT MAP OF ERRORS FOUND WITH SHARED METHODS IN BOTH CASES.

It is apparent from the heat map in Figure 1 that the testing method simulation kit with reduced vision (M2) identified most issues in both case studies. This method also found most

cognitive issues, which might be unexpected at first glance. However, many cognitive issues are linked to poor descriptions or lack of explanations, and this often makes it even harder for the person to understand the purpose of a particular step or page. The WCAG testing method (M6) also found many cognitive issues, but had more trouble finding critical issues. This is most likely because most of the WCAG criteria are high level and too general.

A high coverage of critical issues was found by the screen reader testing method (M4) and the VATLab checklist (M7) in both cases, and this is not unexpected since both try to find problems with screen reader support. As both test cases show, if a solution is not created with screen readers in mind, then the solution is usually not possible to use properly with a screen reader and thus many critical issues are identified.

The high contrast testing method (M5) found generally few issues, and also few critical or cognitive. It is worth mentioning that most of the issues found with high contrast were also found by other methods (over 75%).

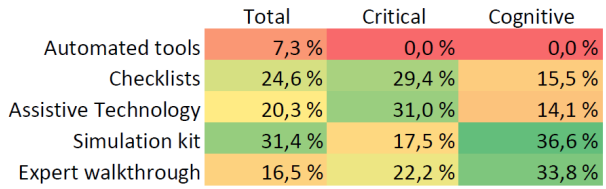

FIGURE 2. HEAT MAP OF ERRORS FOUND WITH THE VARIOUS TESTING METHOD GROUPS.

Figure 2 shows a heat map of testing method groups for both cases. The heat map supports the notion that some testing method groups are better at finding critical issues, while others are better at finding cognitive issues. As can be seen from Figure 2 both simulation kit and expert walkthrough have a high discovery rate for cognitive issues, while checklists and assistive technology testing methods have a high discovery rate for critical issues.

D. Testing time usage

During the evaluations, we noted the time used per tester, and the average time per testing method can be found in Table IX. The numbers in Table IX is the average for both Case A and B, and as the table shows, most methods use between 20 and 60 minutes, with the exception of method 3 and 5. The reason for such low numbers for M3 is because the solution was not very depended on dexterity as explained in Section V. Method M5 also gave very quick test times, mostly because it was fairly obvious what was standing out and what was difficult to see with a high contrast mode.

The used time recorded is depending both on scenario, experience and solution that is being tested. Some testing methods were not tested with scenarios, like automated checker (M1), WCAG (M6) or VATLab checklist (M7), but used in an open exploratory testing approach. We saw on average higher testing times for most methods in Case A compared to Case B, and this is most likely because Case A was more complex with more pages than Case B. However, the average time per testing method is representative for both cases. More important is that the relative difference between the methods were similar for both cases.

TABLE IX. AVERAGE TIME PER METHOD.

Method	*Time*	*Issues Case A*	*Issues Case B*
M1	≈ 20 min	-	27
M2	≈ 45 min	54	58
M3	≈ 10 min	4	-
M4	≈ 30 min	33	16
M5	≈ 10 min	10	16
M6	≈ 60 min	32	25
M7	≈ 45 min	19	15
M8	≈ 60 min	34	-
M9	≈ 60 min	27	-

It is interesting to see the number of issues found compared to average time used per method. This is shown in the two columns to the right in Table IX. In general, more issues are found with a longer average testing time per method, but there are exceptions like the screen reader testing method (M4) that finds many issues in a short amount of time. This is also very apparent for the automated checker (M1) which finds many issues in a very short amount of time. However, most of those issues are more trivial as explained in the previous sections.

VI. COST BENEFIT ANALYSIS

Based on the evaluation results in Section V, we performed a cost-benefit analysis (CBA) of which combinations of accessibility testing methods that identified most issues with regards to resources and knowledge. The motivation for doing a CBA is to get a more objective evaluation of the different testing methods, so it is easier to decide when to include a testing method in the process. CBA is a systematic approach for comparing strengths and weaknesses for different options [40]. It is a well-known technique used in many fields [41], but to our knowledge we are the only ones that have used it for comparing accessibility testing methods [1].

In order to do a CBA we must first create a cost function. We have defined the cost function to be the product of resources and knowledge (explained in III) where $resources$ and $knowledge \in \{1, 2, 3\}$ and *low* corresponds to 1, *medium* to 2 and *high* to 3. This makes sense since both variables contribute equally to the cost of executing a testing method. We argue that the most beneficial accessibility testing methods are those that find a high number of issues, and also many critical and cognitive issues. We can then define the benefit function as the sum of found, critical and cognitive issues. Based on the definition of cost and benefit we can then define the cost-benefit relationship accordingly:

$$CB = \frac{1}{\sqrt{n}} \frac{total^2 + critical^2 + cognitive^2}{resources \times knowledge} \quad (1)$$

Where $total$ is the total number of issues for n methods, $cognitive$ is the total number of cognitive issues for n method, $critical$ is the total number of critical issues for n method and n is the number of methods. We have included squared

TABLE X. COST BENEFIT RESULTS CASE A.

#	*Methods*	*Cost*	*Issues*	*Critical*	*Cognitive*
1	M2, M4, M6, M7	6	64.8%	66.7%	36.8%
2	M2, M4, M6	5	55.9%	49.0%	36.8%
3	M2, M6	3	40.4%	21.9%	36.8%
4	M2, M6, M7	4	49.3%	39.6%	36.8%
5	M2, M4, M6-M8	9	80.8%	82.3%	76.3%
6	M2, M4, M6, M8	8	71.8%	64.6%	76.3%
7	M2, M4, M5-M7	7	69.4%	70.8%	36.8%
8	M2, M4, M6-M9	12	93.4%	95.8%	100%
9	M2, M4, M7	5	49.8%	59.4%	23.7%
10	M2, M4, M6, M7, M9	9	77.5%	80.0%	60.5%
...					
20	M2, M4-M9	13	98.1%	100.0%	100.0%
...					
52	M2-M9	15	100.0%	100.0%	100.0%
...					
254	M3, M5	3	6.6%	4.2%	0.0%
255	M3	2	1.9%	0.0%	0.0%

TABLE XI. COST BENEFIT RESULTS CASE B.

#	*Methods*	*Cost*	*Issues*	*Critical*	*Cognitive*
1	M2	2	36.9%	26.6%	51.5%
2	M1, M2	3	54.1%	26.6%	51.5%
3	M1, M2, M5	4	64.3%	33.3%	66.6%
4	M1, M2, M6	5	70.0%	43.3%	69.7%
5	M2, M5	3	47.1%	33.3%	66.6%
6	M1, M2, M5, M6	6	80.3%	50.0%	84.8%
7	M2, M5	4	52.9%	43.3%	69.7%
8	M1, M2, M4	5	64.3%	50.0%	66.6%
9	M2, M5, M6	5	63.1%	50.0%	84.8%
10	M1, M2, M4, M5	6	74.5%	56.6%	81.9%
...					
18	M1, M2, M4-M7	10	100.0%	100.0%	100.0%
...					

weighting of both cognitive and critical issues since we argue that these issues are more important to discover than minor issues. We used $\sqrt{n}$ instead of n as a penalty for the number of evaluations, since using only n gave a too big penalty when using multiple methods.

A. Analysis from Case A

For Case A we calculated *CB* for all permutations of the different accessibility testing methods to identify the combinations that give most benefit compared to cost. The top results in addition to some selected results are shown in Table X, ordered by *CB*.

Combining all methods (except M3) gives a very high coverage (almost 100%), but comes at a high cost, as shown with #20. The *CB* found 19 better alternatives when considering the costs. The optimal combination of methods that maximize benefit compared to cost is using methods M7, M6, M4 and M2 (#1). This combination has a relatively low cost and discovered almost 65% of all issues in addition to a high number of both critical and cognitive issues (66.7% and 36.8%).

It is not surprising that if more methods are combined then the results are better, but at a higher cost, as for instance shown with combination #5 and #8 in Table X. However, a combination of two methods (#3) gives reasonable good results of discovering around 40% of the known issues, and a large number of both critical and cognitive issues (21.9% and 36.8%).

With a small increase in cost using three testing methods, around 50% of all issues were discovered, as shown with combination #2 and #4. Combination #2 finds 55.9% of all issues, and almost half the known critical issues. It is also worth noting that this testing method combination uses three different accessibility testing method groups (simulation kit, assistive technology and checklist) to discover many different issues.

The first method combination that discover over 80% of known issues is combination #5. This is also the first combination where the persona testing method (M8, M9) is included, but the consequence is high cost. This is the first method combination that requires expert knowledge by including the persona testing method. It is impossible to achieve a very high discovery rate for Case A without including one of the persona testing evaluations, but the downside is that the cost for persona testing is much higher than the rest.

As shown in Table VII, the method simulation kit with reduced dexterity (M3) reported a low number of issues, and this is why there are few combinations in Table X that includes M3. The first method that includes M3 is #53, and here all the methods are included. The two last combinations (#254 and #255) are also based on M3, and this clearly shows that testing method M3 was not very useful for accessibility testing in Case A.

B. Analysis from Case B

For Case B we also calculated *CB* for all testing method permutations to identify the combinations that gives most benefit compared to cost. The top 10 results in addition to the result which use all the methods are shown in Table XI ordered by *CB*.

The optimal approach based on Case B is to use only method M2, which is the disability kit with reduced vision. This has a very low cost, but does not give a high discovery of issues (36.9%). By including more methods, the discovery of issues increases, but naturally so does the cost, as in Case A. It is not until combination #18 that all methods are combined to give a 100% coverage at a high cost.

If only two methods are combined as shown in combination #2, over half the known issues are found (54.1%), but few critical issues are identified. The best combination to find over half the critical issues is with combination #6, where four testing methods are combined to discover 80.3% of known issues and 50.0% of known critical issues. Note that all top 10 results discover at least 51.5% of known cognitive issues.

VATLab checklist (M7) is not part of any top 10 combinations, and this is because of a low ratio of found issues versus cost. This is very different from Case A where the M7 method was part of top 10 multiple times. However, in both Case A and Case B almost the same number of issues were identified, but in Case A many more critical issues were identified compared to Case B.

From Table XI we can also see that the general cost is much lower compared to Table X, and this is because persona testing

methods (M8, M9) were not part of Case B. Persona testing has a very high cost in terms of knowledge requirements, and that is the main reason for higher costs in Case A.

C. Comparison of both cases

From X and XI we see that some methods are prioritized in both top 10 results. Both simulation kit with reduced vision (M2), WCAG (M6) and screen reader (M4) testing methods are the most popular testing methods according to the *CB* analysis. Two testing methods are a little more difficult to compare since we have upgraded the knowledge requirement from Case A to B. This includes WCAG (M6) and VATLab checklist (M7) as indicated in Table V. Because of this, we have calculated the *CB* for Case A with the updated knowledge requirement values from Table V to get better comparison data.

In Table XII the best combination for Case A and Case B is presented. The updated calculation for Case A is also included (A updated). The best combination is determined by counting the occurrences of the testing methods in the top 10 results. In general all methods are part of the best combinations, with the exception of simulation kit with reduced dexterity (M3), but the ordering is a little different depending on the original Case A or the updated Case A.

TABLE XII. COMBINATION OF CASE A AND B.

Case	*Best combination*
A + B	M2, M6, M4, M7, M1/M5, M8, M9
A updated + B	M2, M4/M6, M1/M5/M8, M7, M9

Simulation kit with reduced vision (M2) is the clear winner in both case studies, while WCAG (M6) and screen reader (M4) battle for second place. The other testing methods (M1, M5, M7, M8) compete for the remaining 4th to 7th places while M9 is in the last place for both A and B.

VII. DISCUSSION

Based on the results in Section VI we found that the combination of several methods gives good results compared to the investment. Our CBA from Case A and Case B shows that the testing methods simulation kit with reduced vision, WCAG and screen reader gives good results in both cases with a moderate cost, and yet, the combination of these methods discover a large number of issues with the solutions we evaluated. In addition, the automated checker from Case B showed good issue discovery with very little resource and knowledge requirements, while persona testing from Case A showed great issue discovery, particular for finding cognitive issues, but with high knowledge requirement. Based on an overall assessment from both cases, these five methods gives the best combination of methods. However, the results do not say anything about when to apply the different methods during a development process.

In Figure 3 we have illustrated how to prioritize the different accessibility testing methods in an agile development process, and we call this the *agile accessibility spiral* [1]. The circular layers represent the testing methods, and start from the center with simulation kit using reduced vision and expand outwards to show the priority of the testing methods. The outer layers illustrate an agile process that covers four common activities (design, development, test, review) in successive iterations. The activities are not necessarily clearly separated as shown in the illustration, and they often happen in parallel. The motivation behind the *agile accessibility spiral* is that the different testing methods can be included in all activities in an agile process. The total cost increases as more testing methods are included during testing, but the number of issues discovered also increases, as indicated with the spiral arrow circling outwards from the center.

We have primarily based the ordering of the accessibility testing methods in the spiral on Table XII, but with some exceptions. Testing accessibility using simulation kit with reduced vision is the only method which is always part of top ten results in both cases, and is thus a natural first choice as a testing method. Not only does the method discover most issues, but it is also low on knowledge requirements. Furhermore, the method is exciting to use according to the testers that were part of both studies. Or, as one designer said:

> "I greatly appreciated using the glasses because they made me *feel* the error."

The statement is also supported by the results in Section VI. Finally, the simulation kit for reduced vision is a method that can be performed many times without affecting the bias of the tester, since it is a wearable gadget that is used by the tester and not so much a mental testing approach.

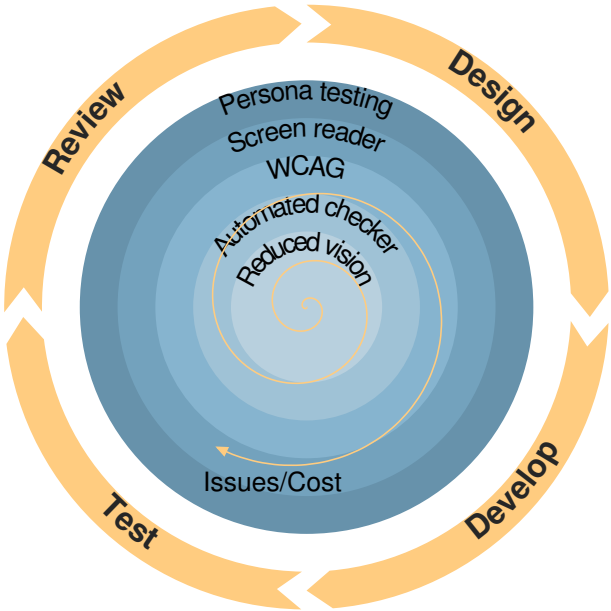

FIGURE 3. AGILE ACCESSIBILITY SPIRAL.

The second accessibility testing method that should be used is the automated checker, since this require very little investment or prior knowledge. Some automated checkers can also be part of the developer IDE or build cycle, and we argue that using an automated checker should be prioritized after simulating reduced vision. Automated checker was only part

of the top 10 results 6 times, but both the low average testing time and low requirements (resources and knowledge) are strong indications that the testing method could be done before both the WCAG and screen reader testing methods. Feedback from the tester during the case studies support that automated checker is something they prefer over most methods, and one developer summarized the first two testing method in this manner:

> "I think the automated checker revealed most errors, and I liked that tool the most because it was quick and easy to use. For example, it quickly revealed the high number of contrast errors. That said, it was not until I had the glasses on that I actually understood how severe the errors were. So maybe it is best to use WAVE to reveal the errors and then the glasses to feel them on the body."

Checklists like WCAG is part of the top 10 a total of 13 times, and we therefore place this method as the third testing method in the *agile accessibility spiral*. It should be noted that VATLab checklist was part of top 10 a total of 3 times, but the method had much overlap with other methods (over 75%), and we therefore decided to drop the VATLab checklist from the *agile accessibility spiral*. We argue that WCAG covers most of the VATLab criteria, but discovers more issues and is thus a natural choice. In addition, WCAG is an international standard and legislation that most developers and testers are familiar with.

However, we found that the use of WCAG and VATLAB checklists were perceived as tedious and somewhat difficult, and this is another reason why we place WCAG as the third method instead of the second testing method. One developer commented:

> "I found the WCAG guidelines cumbersome to use. The success criteria was not very explanatory which made it hard for me to take an objective decision whether the criteria was fulfilled or not. I also found it too time-consuming, I would much rather use automated tests or glasses."

Our findings are similar to Farrelly [42], who found that WCAG was perceived by web developers as overwhelming, confusing and with an obtuse and too technical language. While Freire et al. [22] found in their study that nearly 40% of the web developers did not have any knowledge about the WCAG, our findings suggest that developers have knowledge of the guidelines, but that it is too difficult and tiresome to use them regularly.

Screen reader is part of top ten a total of 11 times, and is an obvious choice as the fourth testing method. As Figure 2 shows, this method discovers the highest amount of critical errors of all the methods. The last testing method in the spiral is persona testing, which should be performed less often since the cost is quite high, but also because it is a mental process which might be biased if performed too often by the same person. However, if the team has experience with persona testing or is willing to invest in training, then persona testing is a very good accessibility testing method that we strongly recommend.

We have not included more methods in Figure 3, since these five methods cover over 82.7% of known issues as shown in Table VII and Table VIII. As a minimum at least two different testing method should be included during testing [43].

A. Use of the testing methods

We argue that it is important to do accessibility testing at early stages in the development of a user story to avoid costly adjustment at a later stage [7]. It is well worth the investment to make sure that a solution works well for both screen readers and for people with reduced vision, since this will benefit all end-users [9].

During the up-front analysis and design, sketches and wire-frames are created by hand or by tools. The motivation of making these sketches is to create something visual that can be discussed and explained to developers, testers and product owners. There are limited accessibility testing methods that can be used on such sketches, but we argue that both simulation kit with reduced vision and high contrast testing method may be used with success. It will, of course, depend on the quality of the sketches, but bad contrast and use of too small font sizes are typical issues that will be discovered with these methods. Besides, some elements of the WCAG checklist can be used for assessing logic, flow and common general errors. One designer expressed interest for doing accessibility testing early on:

> "I would probably have used the glasses once a week, and I would also have used them early in the projects. For example, if I were deciding on fonts and colours I would put the glasses on to check how someone with reduced vision would experience the solution."

Once a prototype is implemented, then more testing methods can be used naturally. The idea behind a prototype is to verify concepts of sketches and maybe try different interactions or ideas. In addition to the testing methods that can be applied during the design, elements from VATLab checklist and persona testing can be used. It depends on how good the prototype is, but some parts of the VATLab checklist can be used to test for screen reader support, and persona testing can be used to identify cognitive issues already in the prototype stage. An automated checker can be used at a prototype, depending on the product, to detect design issues like poor color contrast.

While simulation kits for reduced vision can be used early in the development and testing of a user story, other testing methods,for example screen readers and persona testing, need a more stable version of the code in place before the methods can be used successfully. The reason for this is that screen readers require elements to be marked so that the screen readers can find the required information. Persona testing is most suitable at the end of an implementation of a user story, since entire features should have been implemented for this

testing method, and the testers should use the real software product.

B. Implications for practice

Our findings have implications for software organizations wanting to build more usable and accessible software. Practitioners can use the agile accessibility spiral to assess the appropriate tools to test accessibility early and frequently in the agile development cycle.

While there exist many accessibility testing methods, they are not that known to agile team members. Companies should strive to inform their employees about what techniques they can use and how to use them during the development cycle. A survey on accessibility awareness among web development project members [22] finds that the main reasons for not considering accessibility issues in the development process were the lack of formal requirements from the organization, lack of customer requirements and lack of training.

Based on the evaluation results and the cost-benefit analysis, we strongly suggest that all agile software projects buy glasses for simulating reduced vision and make them easily available for their employees to get hold of. Using the glasses was undoubtedly the tool that revealed the most errors, and additionally it was an eye-opener for many of the developers. However, while all participants claimed they would use the glasses if they had them at their desk, they said they would not order them themselves. This barrier for use must be removed by the project leaders. Both designers and developers should use automated checkers, such as WAVE, as early as possible in the projects and as often as possible. Use of automated checkers to quickly get an overview of errors and then using the glasses to get a feeling of how the errors would be perceived by a person with reduced vision creates new insights and the iterative use of these two methods complement each other.

In order to make software as accessible as possible, all members of an agile software team should have basic knowledge of accessibility testing and universal design, and norms and practices should be incorporated into the development process to ensure that the software developed is accessible. Norms are an integral aspect of how agile teams work [44], and promoting the right norms in the team may influence how developers think of accessibility testing. For example, promoting norms that simulation kits are frequently used and that accessibility issues are discussed in team meetings.

Our findings suggest that checking for accessibility issues with the use of automated checkers and simulation kits will make developers and designers in agile projects more positive towards continuous accessibility testing than the use of manual guidelines. Use of guidelines and checklists was perceived as tedious and boring.

VIII. Limitations

We only covered a limited selection of possible methods and tools for accessibility testing in our study, and other methods could prove to be more beneficial. We have, however, selected methods from a wide range with respect to what they cover of technical and usable accessibility.

The cost-benefit analysis was calculated based on the requirements in resources and knowledge of each method. The classification of methods as low, medium or high in demands for resource and knowledge could be further validated. One should also consider other factors contributing to cost in the analysis. For example, the amount of time spent in performing a test method is a cost. Besides, satisfaction and ease of use could also be possible beneficial criteria to include.

The categorization of issues as critical or cognitive is subjectively performed by the authors and future research should try to standardize the classification of issues. However, each evaluation was observed and reported by at least two, sometimes three, observers and most of the testing methods where used in both case studies. These conditions should result in a reduced number of errors.

Our study of accessibility testing methods was limited in number of participants and the size of the evaluated applications. Hence, future work should explore the different accessibility testing methods for other software solutions. Additionally, more studies should be performed to validate the cost-benefit relationships and the *agile accessibility spiral*.

IX. Conclusion

In this study, we investigated methods for accessibility testing in agile projects. Based on evaluations from two case studies and a cost-benefit analysis we proposed the *agile accessibility spiral*. We included five of the tested methods in the spiral. The cost and knowledge of testing methods increase from the center of the spiral and outwards, but the discovery of issues also increases when moving outwards from the center. We recommend to choose methods from the center and gradually apply more testing methods.

The different testing methods should be adjusted to the software solution and the expertise of the team, so they fit into the agile process. The more knowledge and experience an agile team gains, the smaller the circles in the *agile accessibility spiral* will become. We argue that developers and testers can contribute more with accessibility testing to deliver a better end product as shown with our two case studies. We also believe they will be more positive towards testing for accessibility with the use of simulation kits for reduced vision and automated checkers instead of only using guidelines such as WCAG.

More research on evaluations of accessibility testing should be carried out, and it would be interesting to evaluate other testing methods than those used in this study and incorporate them into the *agile accessibility spiral*. Further work should also investigate whether other factors such as time, satisfaction and ease of use should be included in the cost-benefit analysis.

References

[1] A. Bai, H. C. Mork, and V. Stray, "A Cost-benefit Evaluation of Accessibility Testing in Agile Software Development," in ICSEA 2016 : The Eleventh International Conference on Software Engineering Advances, 2016, pp. 62–67.

[2] United Nations. Convention on the Rights of Persons with Disabilities. [Online]. Available: https://www.un.org/development/desa/disabilities/convention-on-the-rights-of-persons-with-disabilities.html [Accessed: 2017-05-26].

[3] H. Petrie and N. Bevan, "The evaluation of accessibility, usability and user experience," The universal access handbook, 2009, pp. 10–20.

[4] R. Mace, "What is universal design," The Center for Universal Design at North Carolina State University. Retrieved Retrieved, vol. 19, 1997, p. 2004.

[5] J. Nielsen, "Return on investment for usability," Jakob Nielsen's Alertbox, January, vol. 7, 2003.

[6] L. C. Cheng and M. Mustafa, "A reference to usability inspection methods," in International Colloquium of Art and Design Education Research (i-CADER 2014). Springer, 2015, pp. 407–419.

[7] M.-L. Sánchez-Gordón and L. Moreno, "Toward an integration of web accessibility into testing processes," Procedia Computer Science, vol. 27, 2014, pp. 281–291.

[8] B. Haskins, B. Dick, J. Stecklein, R. Lovell, G. Moroney, and J. Dabney, "Error Cost Escalation Through the Project Life Cycle," in Incose - Annual Conference Symposium Proceedings- Cd Rom Edition; 2004, 2004.

[9] F. Paz and J. A. Pow-Sang, "A systematic mapping review of usability evaluation methods for software development process," International Journal of Software Engineering and Its Applications, vol. 10, no. 1, 2016, pp. 165–178.

[10] T. Dingsøyr, S. Nerur, V. Balijepally, and N. B. Moe, "A decade of agile methodologies: Towards explaining agile software development," Journal of Systems and Software, vol. 85, no. 6, 2012, pp. 1213 – 1221.

[11] V. Stray, N. B. Moe, and G. R. Bergersen, "Are daily stand-up meetings valuable? A survey of developers in software teams," in Agile Processes in Software Engineering and Extreme Programming: 18th International Conference, XP 2017, Cologne, Germany, May 22-26, 2017, Proceedings, H. Baumeister, H. Lichter, and M. Riebisch, Eds. Cham: Springer International Publishing, 2017, pp. 274–281. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-57633-6_20 [Accessed: 2017-06-01].

[12] K. Schwaber and M. Beedle, "Agile Software Development with Scrum". Upper Saddle River, NJ: Prentice Hall, 2002.

[13] V. Stray, D. I. K. Sjøberg, and T. Dybå, "The daily stand-up meeting: A grounded theory study," Journal of Systems and Software, vol. 85, 2016, pp. 101 – 124.

[14] J. Highsmith and A. Cockburn, "Agile software development: The business of innovation," Computer, vol. 34, no. 9, 2002, pp. 120 – 127.

[15] G. Jurca, T. D. Hellmann, and F. Maurer, "Integrating agile and user-centered design: a systematic mapping and review of evaluation and validation studies of agile-ux," in Agile Conference (AGILE), 2014. IEEE, 2014, pp. 24–32.

[16] G. Zimmermann and G. Vanderheiden, "Accessible design and testing in the application development process: considerations for an integrated approach," Universal Access in the Information Society, vol. 7, no. 1-2, 2008, pp. 117–128.

[17] J. C. Lee and D. S. McCrickard, "Towards extreme (ly) usable software: Exploring tensions between usability and agile software development," in Agile Conference (AGILE), 2007. IEEE, 2007, pp. 59–71.

[18] D. Salah, R. F. Paige, and P. Cairns, "A systematic literature review for agile development processes and user centred design integration," in Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, ser. EASE '14. New York, NY, USA: ACM, 2014, pp. 5:1–5:10. [Online]. Available: http://doi.acm.org/10.1145/2601248.2601276 [Accessed: 2017-03-01].

[19] R. Bonacin, M. C. C. Baranauskas, and M. A. Rodrigues, "An agile process model for inclusive software development," in Enterprise information systems. Springer, 2009, pp. 807–818.

[20] S. Horton and D. Sloan, "Accessibility in practice: a process-driven approach to accessibility," in Inclusive Designing. Springer, 2014, pp. 105–115.

[21] T. Halbach and K. Fuglerud, "On assessing the costs and benefits of universal design of ict." Studies in health technology and informatics, vol. 229, 2016, p. 662.

[22] A. P. Freire, C. M. Russo, and R. P. M. Fortes, "A survey on the accessibility awareness of people involved in web development projects in Brazil," in Proceedings of the 2008 international cross-disciplinary conference on Web accessibility. ACM, 2008, pp. 87–96.

[23] C. Paddison and P. Englefield, "Applying heuristics to accessibility inspections," in Interacting with Computers, vol. 16, no. 3, 2004, pp. 507–521.

[24] Web accessibility evaluation tools list. [Online]. Available: https://www.w3.org/WAI/ER/tools/ [Accessed: 2017-02-01].

[25] NetBeans. Accessibility Checker. [Online]. Available: http://plugins.netbeans.org/plugin/7577/accessibility-checker [Accessed: 2017-05-26].

[26] C. Cardoso and P. J. Clarkson, "Simulation in user-centred design: helping designers to empathise with atypical users," Journal of Engineering Design, vol. 23, no. 1, 2012, pp. 1–22.

[27] A. M. Silverman, J. D. Gwinn, and L. Van Boven, "Stumbling in their shoes disability simulations reduce judged capabilities of disabled people," Social Psychological and Personality Science, vol. 6, no. 4, 2015, pp. 464–471.

[28] S. L. Henry, "Just ask: integrating accessibility throughout design". Lulu. com, 2007.

[29] W3C. Web Content Accessibility Guidelines. [Online]. Available: https://www.w3.org/TR/WCAG20/ [Accessed: 2017-03-01].

[30] T. Schulz and K. S. Fuglerud, "Creating Personas with Disabilities," in Computers Helping People with Special Needs, ser. Lecture Notes in Computer Science, K. Miesenberger, A. Karshmer, P. Penaz, and W. Zagler, Eds., vol. 7383. Linz, Austria: Springer Berlin / Heidelberg, 2012, pp. 145–152.

[31] A. Bai, H. C. Mork, T. Schulz, and K. S. Fuglerud, "Evaluation of accessibility testing methods. which methods uncover what type of problems?" Studies in health technology and informatics, vol. 229, 2016, p. 506.

[32] J. S. Dumas and J. Redish, "A practical guide to usability testing". Intellect Books, 1999.

[33] R. G. Bias and D. J. Mayhew, "Cost-justifying usability: An update for the Internet age". Elsevier, 2005.

[34] Wave web accessibility evaluation tool. [Online]. Available: http://wave.webaim.org/ [Accessed: 2017-02-01].

[35] K. S. Fuglerud, S. E. Skotkjerra, and T. Halbach. (2015) Håndbok i testing av websider med hjelpe-middel-program-vare, Virtuell hjelpe-middellab.

[36] Cambridge. Inclusive Design Toolkit. [Online]. Available: http://www.inclusivedesigntoolkit.com [Accessed: 2017-02-01].

[37] NV Access. NVDA Screen Reader. [Online]. Available: https://www.nvaccess.org/ [Accessed: 2017-02-01].

[38] Futureid. [Online]. Available: http://www.futureid.eu/ [Accessed: 2017-03-01].

[39] Norsk pensjon. [Online]. Available: https://www.norskpensjon.no/ [Accessed: 2017-03-01].

[40] M. M. Mantei and T. J. Teorey, "Cost/benefit analysis for incorporating human factors in the software lifecycle," Communications of the ACM, vol. 31, no. 4, 1988, pp. 428–439.

[41] A. E. Boardman, D. H. Greenberg, A. R. Vining, and D. L. Weimer, "Cost-benefit analysis: concepts and practice," 2006.

[42] G. Farrelly, "Practitioner barriers to diffusion and implementation of web accessibility," Technology and Disability, vol. 23, no. 4, 2011, pp. 223–232.

[43] K. S. Fuglerud, "Inclusive design of ICT: The challenge of diversity". Dissertation for the Degree of PhD, University of Oslo, Faculty of Humanitites, 2014.

[44] V. Stray, T. E. Fægri, and N. B. Moe, "Exploring norms in agile software teams," in Proceedings of the International Conference on Product-Focused Software Process Improvement. Springer International Publishing, 2016, pp. 458–467.