A Cost-benefit Evaluation of Accessibility Testing in Agile Software Development

Aleksander Bai, Heidi Camilla Mork
Norwegian Computing Center, Oslo, Norway
Email: {aleksander.bai|heidi.mork}@nr.no

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

***Abstract*—Accessibility testing in software development is testing the software to ensure that it is usable by as many people as possible, independent of their capabilities. Few guidelines exist on how to include 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 four 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 the optimal combinations 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

The past decades 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. Accessibility focuses on letting people with the widest range of capabilities be able to use a product or service [1]. There is an increased focus on accessibility from governments and the United Nations with "Convention on the Rights of Persons with Disabilities" [2].

Studies show that doing usability testing is costly and can take around 8-13% of the project's total budget [3]. Much of the cost goes to recruiting participants and evaluators in addition to the man-hours required for conducting and evaluating the results [4]. 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 [5] [6]. Many studies show that software that is hard to use, or have features that are hard to understand, make users find better alternatives [7]. There might also be legal requirements to provide accessibility.

We argue that developers and testers in 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. Our approach is targeted towards agile software development, since it has become the mainstream development methodology [8] [9].

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 [5]. 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 approach and the issues that were found during the evaluations. Section V reports our cost-benefit analysis of the accessibility testing methods and Section VI discusses the results. Finally, we summarize and conclude in Section VII.

II. Related Work

Zimmermann and Vanderheiden [10] 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 [11], and in recent years, there has been an increasing interest in Agile UX (User Experience) [12]. Bonacin et al. [13] 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.

A recent systematic review of usability testing methods for software development processes [7] request 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.

 ISBN: 978-1-61208-498-5

III. BACKGROUND

There are several methods for testing usability [7] and accessibility [10] in software development: automated tools, guidelines, expert walkthrough, interviews and user testing to name a few. There are different alternatives of grouping these methods [14], and we have chosen to divide them into five groups based on the resources and knowledge required when using the methods in software development, as shown in Table TableI.

The amount of resources and knowledge required is categorized as either *low*, *medium* or *high*. 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 considerably 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.

TABLE I. ACCESSIBILITY TESTING GROUPS.

#	*Group*	*Resource requirements*	*Knowledge requirements*
1	Automated tools	Low	Low
2	Checklist and guidelines	Low	Low
3	Simulation using wearables	Medium	Low
4	Expert walkthrough	Low	High
5	User testing	High	Medium

Automated testing tools require fewer resources in terms of time, knowledge and resources, compared to other testing methods. It is quite feasible for a developer or tester to install a tool and run an evaluation, and most of the tools are also free to use. There are numerous alternatives out there, like the NetBeans accessibility module [15] that integrate directly into the developer's tools, but most automated tools only support simulation of visual impairments or a limited variant of other impairments. Overall, the automated testing tools are *low* in resources required to acquire and use them. The automated tools usually explain the evaluation results in great details to the operator, and give suggestions on how to fix or improve the problems that have been found. This means very little prior knowledge is required from the operator, and explains why we have also labeled the automated tools methods with *low* knowledge requirements in Table I.

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 [16] is a common choice. It is easy to find both checklists and guidelines on the Internet, and they have good and detailed documentation on how to perform the evaluation and how to assess the results from the evaluation. This is why we have labeled checklists and guidelines methods with *low* for resource and knowledge requirements. Even though these methods require little resources and knowledge, studies have shown that guidelines are hard to understand and follow, and have not increased accessibility as much as anticipated [17].

There are many different tools or wearables that an able-bodied person can use in a simulation. 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 [18]. It is important to note that the intention of a simulation kit is not to simulate the disability in itself, which is highly criticized [19]. Simulation kits are fairly cheap to purchase and setup, but it requires some planning; the simulation kits must be evaluated to discover which is most suitable, and the simulation kits must also be purchased. The planning and cost aspect is the reason for labeling the simulation methods as *medium*. However, simulation kits come with good instructions on how to operator them and normally requires no prior knowledge, which is reflected by *low* knowledge requirements in Table I.

Expert walkthrough, also called persona walkthrough or persona testing approach [20], 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. The approach 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. All expert walkthrough methods requires expert knowledge (as the name indicates) and is thus marked with *high* knowledge requirements. However, there are few resource requirements for expert walkthrough methods, and this is why they are labeled with *low* for resource requirements.

The best approach for accessibility testing is user testing, since the actual users are involved and the testers does not have to do any approximation of impairments or mental states [21] [22]. However, it is also an expensive method because it requires much planning, recruitment and management [4]. Examples of user testing involves inquiry, interview, focus group and questionnaire. This means that resource requirements are *high* as indicated in Table I. User testing methods requires some prior knowledge on how to recruit, organize and conduct user testing, and we have indicated this with *medium* knowledge requirements.

IV. ACCESSIBILITY EVALUATIONS

We conducted eight evaluations. The goal of the evaluations was to investigate what kind of issues the different test methods can discover, and how the test methods differ from each other. The findings of the evaluations were then used in a cost-benefit analysis, to suggest where in an agile development process the methods might be most valuable.

The system used for evaluation was a pilot for electronic identification, developed during the EU project FutureID. The pilot uses a software certificate or an ID card with a card reader for the authentication process. The pilot uses both a

 ISBN: 978-1-61208-498-5

Java client and a web front end, and consists of around ten different user interfaces with varying complexity.

A. Selected Methods

We selected methods from the groups in Table I suitable to be performed by a person working in an agile team, i.e., from all groups except user testing and testing with automated tools. Table II shows the selected method types; Simulation kit (group 3), VATlab (group 2), Persona (group 4) and Manual WCAG (group 2).

The four types of methods were selected based on a combination of resources and knowledge required to perform a method. Ideally, in a development cycle, one wants to use as little resources as possible on accessibility testing, but at the same time discover the most critical accessibility issues that exists in the software. Therefore, we focused on testing methods that are relatively inexpensive to conduct in terms of time and resources. This is why user testing was omitted, since it is an expensive method to conduct. Automated tools were also omitted, since they are limited in what they can actually test.

Almost all methods are labelled as *low* with regards to resources. The only method with prerequisites is the simulation kit 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 3 involves using a screen reader that requires knowledge on how to operate it, but almost everyone can learn how to use a screen reader in a short 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 of them. The persona testing methods requires much more prior knowledge, both in terms of the method itself, but also on the impairments that is being play-acted. Thus, the persona testing methods are *high* in knowledge demands.

TABLE II. OVERVIEW OF THE EIGHT EVALUATIONS

#	*Method*	*Impairments*	*Resources*	*Knowledge*
1	Simulation kit	Reduced vision	Medium	Low
2	Simulation kit	Reduced dexterity	Medium	Low
3	VATLab	Blindess	Low	Medium
4	VATLab	Light sensitivity	Low	Low
5	VATLab	Multiple	Low	Low
6	Persona	Dyslexia	Low	High
7	Persona	Being old	Low	High
8	Manual WCAG	Multiple	Low	Low

For the simulation kit approach, two impairments were selected that cover both visual and physical dimensions. We used Cambridge inclusive design glasses [23] for simulating reduced vision, and the Cambridge inclusive design gloves [23] to simulate dexterity reduction. These gloves are typically used for testing a physical products, but they were included in our evaluation since there was a card reader involved.

We used a Virtual Assistive Technology Lab (VATLab) to test various assistive technologies [24]. The VATLab contains two different screen readers; NVDA and SuperNova. These tools are used by blind people, and give a good indication of how accessible the solution is for this type of impairment. We also used the built-in high contrast mode in Windows. The VATLab project also includes a checklist for evaluating web pages for screen readers, and we used this checklist in the evaluation [24].

For the persona walkthrough we defined two personas that we had experience with, one being old and one with dyslexia. For each persona there was an expert play-acting the particular persona while performing the predefined test-scenarios. To make it more realistic, the persona testing of being old was also conducted with two layers of Cambridge glasses to simulate reduced vision that often comes with age.

Finally, we conducted a manual testing of the WCAG checklist. The testing was performed with supportive use of available browser plugins to check for instance color contrast.

B. Participants

Six 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.

C. Procedures

Before we started the evaluation, we defined whether an issue was critical or cognitive. 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.

All the evaluations were conducted on the same machine with the same setup to ensure an equal test environment. Each evaluation also 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. All the evaluations were conducted by at least two different participants, and the results were aggregated.

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 system. 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

 ISBN: 978-1-61208-498-5

the listed order to avoid biasing the participants as they should gradually progress a little further in the logging process. Each method took under two hours to complete for a single participant for all the scenarios.

D. Evaluation results

As can be seen from Table III, 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 III. ISSUES FOUND.

Method	*Issues*	*Critical*	*Cognitive*
1	54	14	9
2	4	0	0
3	33	26	0
4	10	4	0
5	19	17	0
6	34	15	15
7	27	13	9
8	32	7	5
	213	96	38

The simulation kit methods found fewer critical issues than VATLab and persona testing, and this is mostly because most of the issues reported were visual problems that were annoying at best and in most cases problematic, but not marked as critical since it did not hindered further progress. Of the critical issues discovered with simulation kit, 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 highly related to the software that is evaluated.

WCAG also reported few critical issues, however 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 VATLab methods since it has a much finer granularity. WCAG also reported few cognitive issues for the same reason.

VATLab methods reported on average the most critical issues, and many of the issues were related to poor compatibility for screen readers. We suspect that more issues could be found if there had not been so many critical problems which made further investigate in many screens impossible.

Persona testing methods found the most cognitive issues, and this is not unexpected since the persona used in the evaluations had focus 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 screen 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.

V. RESULTS

Based on the evaluation results in Section IV, we performed an cost-benefit analysis (CBA) of what combinations of accessibility testing methods that discovered 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 evaluate when to include a testing method in the process. CBA is a systematic approach for comparing strengths and weaknesses for different options [25]. It has not been used for comparing accessibility testing methods to our knowledge, but it is a well known technique that is used in many fields [26].

In order to do a CBA, we first defined the cost to be the combination of resources and knowledge where $resources$ and $knowledge \in \{1, 2, 3\}$ where *low* corresponds to 1, *medium* to 2 and *high* to 3. This makes sense since both variables contributes 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, but also many critical and cognitive issues. We can then define the benefit as the sum of found, critical and cognitive issues. Based on the 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 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.

We calculated *CB* for all permutations of the different accessibility testing methods to identify the combinations that gives most benefit compared to cost. The top results in addition to some selected results are shown in Table IV ordered by *CB*.

Combining all methods (except method 2) 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 5, 8, 3 and 1 (#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 IV. 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%).

 ISBN: 978-1-61208-498-5

TABLE IV. COST BENEFIT RESULTS.

#	*Methods*	*Cost*	*Issues*	*Critical*	*Cognitive*
1	5, 8, 3, 1	6	64.8%	66.7%	36.8%
2	8, 3, 1	5	55.9%	49.0%	36.8%
3	8, 1	3	40.4%	21.9%	36.8%
4	5, 8, 1	4	49.3%	39.6%	36.8%
5	5, 8, 6, 3, 1	9	80.8%	82.3%	76.3%
6	8, 6, 3, 1	8	71.8%	64.6%	76.3%
7	5, 8, 4, 3, 1	7	69.4%	70.8%	36.8%
8	5, 8, 7, 6, 3, 1	12	93.4%	95.8%	100%
9	5, 3, 1	5	49.8%	59.4%	23.7%
10	5, 8, 7, 3, 1	9	77.5%	80.0%	60.5%
...					
20	5, 8, 7, 6, 4, 3, 1	13	98.1%	100.0%	100.0%
...					
53	5, 8, 7, 6, 4, 2, 3, 1	15	100.0%	100.0%	100.0%
...					

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 use three different accessibility testing method types (simulation kit, VATLab and Manual WCAG testing) to discovery many different issues.

VI. DISCUSSION

Based on the results in Table IV we found that the combination of several methods provides good results compared to the investment. Our CBA shows that combining the testing methods 5, 8, 3 and 1 is the most profitable combination of methods. The cost is moderate, and yet, the combination of methods discover a large number of the issues with the software we evaluated. However, the results does not say anything about when to apply the different methods during a development process.

Testing accessibility using simulation kit with reduced vision (method 1) is the only method which is always part of top ten results. Manual WCAG (method 8) is part of the combinations a total of 9 times, while testing blindness using VATLab (method 3) is part of top ten a total of 8 times. VATLab with checklist (method 5) is part of the top 10 results 7 times, while dyslexia persona testing (method 6) is part of the top results three times. Based on these results we can make some general recommendations on which methods to include in accessibility testing during an agile process, and the order in which they should be included.

In Figure 1 we have illustrated how to prioritize the different accessibility testing methods in an agile development process, and we call this the *agile accessibility spiral*. The circular layers represents the testing methods, and start from the center with automated tests and expands outwards to show the priority of the testing methods. The red arrow illustrates an agile process that spirals outwards and cover the same activities in different iterations, and the motivation behind the *agile accessibility spiral* is that the different testing methods can be included in all the activities. The total cost increase as

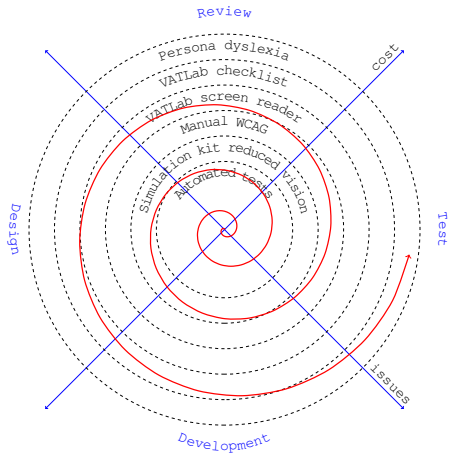

FIGURE 1. AGILE ACCESSIBILITY SPIRAL.

more testing methods are included in testing, but the number of issues discovered also increases.

The different process activities are shown at the edge of the circle with blue separations between the different activities. The activities are gradual and not necessarily clearly separated as shown in the illustration, and they often happen in parallel. We have illustrated four common activities (design, development, test, review) that usually occur in agile software development.

Automated tests are always a vital part of any development process and are thus in the center of the spiral, but we argue that the first accessibility testing method that should be added to automated tests is the simulation kit for reduced vision. This is supported by our results in Table IV, and by the knowledge required as shown in Table I. It is a method that can be performed many times without affecting the bias of the tester too much, since it is a wearable gadget that is used by the tester and not so much a mental testing approach.

Other testing methods like persona testing 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. After the simulation kit method with reduced vision the methods follow successive with manual WCAG, VATLab and blindness, VHL checklist, persona dyslexia. We have not illustrated more methods in Figure 1, since these 5 methods cover over 80% of known issues as shown in Table IV. As a minimum at least two different testing method should be included during testing [27].

During the first iterations in an agile development process, a prototype is often developed, and since the motivation behind a prototype is to show a concept and not necessarily think about all possible outcomes or users, it is still beneficial to do some accessibility testing to avoid costly adjustment at a later stage [5]. However, not all accessibility testing methods are suitable for testing against prototypes or even design sketches. The agile accessibility spiral in Figure 1 also incorporate this in the ordering.

Simulating reduced vision with simulation kit can be done

 ISBN: 978-1-61208-498-5

against both prototype and design sketches, and parts of WCAG can also be tested at an early stage. Further out in the agile accessibility spiral layers, when testing blindness with VATLab and screen readers, a more stable software version should be tested instead of design sketches or prototypes. This is because screen readers requires elements to be marked so the screen readers can find the required information.

VII. Conclusion

Based on our results, we recommend to use the *agile accessibility spiral*, as a reference for accessibility testing in agile development. We recommend to start from the center and gradually apply more testing methods as the project progresses. The cost and knowledge of testing methods increase from the center and outwards, but the discovery of issues also increases when moving outwards from the center. Ideally, if the team has access to staff that can perform persona testing, or are willing to invest the resources to train one or more in persona testing, then we strongly recommend to include persona testing as part of the development cycle.

The different testing methods should be adjusted to the software and the expertise of the development, 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. And as stated before, we strongly recommend to do testing with actual users at some point in the software development. No amount of automated tools, checklist and guidelines, simulation using wearables or expert walkthrough can replace feedback from real users. However, we argue that developers and testers can contribute more with accessibility testing to deliver a better end product.

Our study of accessibility testing methods was limited in number of participants and the size of the evaluated application, hence, future work should explore the different accessibility testing methods for other software solutions. More research on cost benefit evaluation in accessibility testing should be done, and it would be interesting to evaluate more testing methods and place these in the *agile accessibility spiral*.

Acknowledgment

This research was partially funded as part of the FutureID project. The FutureID project is funded by the EU FP7 program (Grant agreement no: 318424).

References

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

[2] United Nations, "Convention on the Rights of Persons with Disabilities," http://www.un.org/disabilities/convention/conventionfull.shtml.

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] D. Bustard, G. Wilkie, and D. Greer, "The maturation of agile software development principles and practice: observations on successive industrial studies in 2010 and 2012," in Engineering of Computer Based Systems (ECBS), 2013 20th IEEE International Conference and Workshops on the. IEEE, 2013, pp. 139–146.

[9] 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.

[10] 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.

[11] 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.

[12] 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: 1. June 2015].

[13] 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.

[14] K. S. Fuglerud and T. H. Røssvoll, "An evaluation of web-based voting usability and accessibility," Universal Access in the Information Society, vol. 11, no. 4, 2012, pp. 359–373.

[15] NetBeans, "Accessibility Checker," http://plugins.netbeans.org/plugin/7577/accessibility-checker.

[16] W3C, "Web Content Accessibility Guidelines," https://www.w3.org/TR/WCAG20/.

[17] C. Power, A. Freire, H. Petrie, and D. Swallow, "Guidelines are only half of the story: accessibility problems encountered by blind users on the web," in Proceedings of the SIGCHI conference on human factors in computing systems. ACM, 2012, pp. 433–442.

[18] 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.

[19] 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.

[20] 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.

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

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

[23] Cambridge, "Inclusive Design Toolkit," http://www.inclusivedesigntoolkit.com.

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

[25] 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.

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

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

 ISBN: 978-1-61208-498-5