RAISING AWARENESS FOR UNIVERSAL DESIGN BY INTEGRATING ACCESSIBILITY TESTING INTO A CONTINUOUS-DEPLOYMENT PROCESS: A LARGE-COMPANY CASE STUDY

Till Halbach[*], Tom Widerøe[‡], Aleksander Bai[*]

[*] Norwegian Computing Center
[‡] Finn AS

Abstract

This work presents a case study for how to integrate automated accessibility testing in a continuous deployment process. The tools axe and Pa11y were successfully integrated for testing of respectively code fragments and graphical user interfaces. After the discussion of relevant considerations, two surveys are presented, carried out before and after the integration, showing that both tools have the anticipated impact: There was an increase of persons that work with accessibility, they spend more time on accessibility-related work, and also the skill level in the various teams got considerably better.

Keywords: Accessibility, A11Y, universal design, testing, checker, software development, integration, automation

1 INTRODUCTION

With a European Accessibility Act (Proposal for a Directive of the European Parliament and of the Council on the Approximation of the Laws, Regulations and Administrative Provisions of the Member States as regards the Accessibility Requirements for Products and Services, 2015) right around the corner and a working Accessibility Directive for websites and mobile applications (Directive (EU) 2016/2102 of the European Parliament and of the Council of 26 October 2016 on the Accessibility of the Websites and Mobile Applications of Public Sector Bodies, 2016), European countries have recently made progress regarding the accessibility of, among others, websites. In Norway, the current situation is that the universal design of ICT solutions has been regulated since 2013 (Regulation for universal design of information and communication technology (ICT) solutions, 2013) and applies to the Web and other areas belonging to both the public and the private sector. The Norwegian authorities are in the process of adapting the national legislation to the European framework.

In the context of ICT, Universal design (UD) refers to the design of ICT solutions such that they can be used by as many people as possible, while (digital) accessibility describes the degree to which a solution can be used by people with impairments. The situation regarding the status quo of web accessibility in Europe is not quite clear. A 2012 EU press release states that "only one third of Europe's 761,000 public sector and government websites are fully accessible" (European Commission, 2012), and a 2015 audit of 1065 European sites gave a compliance rate of 83% (European Internet Inclusion Initiative, 2015), but, as compared to WCAG, these surveys employed relatively few tests, and more recent numbers are missing. The picture is much clearer in Norway, where a recent accessibility audit of 278 public and private websites and almost 100 tests per site show a compliance rate of only 60% (Norwegian Agency for Public Management and eGovernment, 2018). Clearly, implementing accessibility properly in websites is an ongoing effort despite the fact that this has been mandatory for a couple of years now.

Continuous testing is an often followed strategy by today's agile teams to ensure an accessible solution. However, human testing is costly and time-consuming, be it in terms of developer or expert testing, but particularly user testing (Røssvoll & Fuglerud, 2013). This calls for automated tools which are properly integrated in the process of software building. There are, however, many considerations to make and many degrees of freedom for the desired integration, and the purpose of this project was to investigate how an integration can be successfully achieved, what tools are needed or useful along the way, what can be learned from a pilot integration and what are potential pitfalls, and in particular whether automated checkers do increase the developers' awareness for accessibility.

The research question in this work concerns this last aspect: In how far can automated accessibility testing raise a company's awareness for universal design of IT/ICT solutions and increase the degree of digital inclusion? The main contribution of this paper are the discussion of the integration of automatic accessibility checkers in a company's continuous-deployment production, and the analysis of mixed attitudinal / behavioral data gather during two surveys conducted among the participating teams.

The case provider in this project was Finn AS, a Norwegian company established in 2000, currently with the biggest digital marketplace in Norway with 6.3 million unique users every day and over 60 millions pageviews on average per day. Finn also has one of the largest software development departments in Norway with over 140 developers. Since Finn deploys over 1000 times to production every week, it would not be feasible to test manually or conduct user tests before an update release. Finn was therefore very interested in automating accessibility testing as part of the software building process.

The remaining article is straightforward: After a brief discussion of related work and considerations regarding the integration of the new tools, we present the findings from two surveys that verify the project's success. Conclusions are drawn in the end, in combination with an outlook to future work.

2 RELATED WORK

Accessibility testing is an important part of the process to achieve universally designed solutions (Fuglerud, 2014). The effort, however, has to be balanced by factors like effectiveness, efficiency, time consumption, and costs, to name just the most important ones. It has therefore been advised to employ tool-based accessibility checking before the more expensive expert and user testing (Halbach & Fuglerud, 2016; Leitner, Strauss, & Stummer, 2016; Røssvoll & Fuglerud, 2013). For development teams, the goal of automated accessibility testing is the avoidance of time-consuming and repetitive manual testing, which allows the teams to bundle their efforts on delivering flawless and highly usable software (Putnam, Dahman, Rose, Cheng, & Bradford, 2016).

Previous research has discussed how various accessibility testing tools can be integrated into the build process (Bai, Fuglerud, Skjerve, & Halbach, 2018), but this work focuses mostly on manual and expert testing. The amount of time needed for this, however, is a showstopper for accessibility testing in continuous-development environments with frequent deployments (Parnin et al., 2017). Continuous deployment therefore requires fully automated and easily integratable tools.

Manual and (semi-)automated accessibility checkers have been evaluated previously, but the focus has typically been on static tools ("Accessibility tools audit results - Overview - GDS accessibility team," n.d.; Brajnik, 2004; Centeno, Kloos, Fisteus, & Álvarez, 2006; Halbach & Lyszkiewicz, 2015; Petrie, King, Velasco, Gappa, & Nordbrock, 2007; Tanaka & Da Rocha, 2011; Vigo, Brown, & Conway, 2013), i.e., those that are stand-alone or not straightforward to integrate. Some research has considered continuous deployment, though: One particular work has treated accessibility testing in the context of acceptance testing (Watanabe, Fortes, & Dias, 2012). The created tool integrates well in the automated build process and is capable of testing complex user interaction with a webpage, but it cannot be used to validate for instance stylesheets or markup.

The lack of related research shows that automated accessibility testing which is integrated in a continuous-deployment environment is a relatively new area without much previous experience, neither in the research community nor the industry.

3 PRE-INTEGRATION CONSIDERATIONS & INTEGRATION

The website *finn.no* is made up of 350 microservices. The philosophy behind microservices is that many small and independent modules work together to constitute the overall solution (Namiot & Sneps-Sneppe, 2014). Microservices are developed, tested, built, versioned, and deployed independently of each other by dedicated teams. While this has many advantages, it also introduces complex challenges with regard to how single components, the integrated solution, and particularly the end-to-end user interface can be tested accessibility-wise.

Before the integration, many teams had the concern that going from a situation without accessibility testing to a fully integrated process would instantly burden the developers with a huge workload, and it could potentially lead to a complete production/deployment stop, assuming that even a single accessibility issue breaks the build. There are different strategies to meet those concerns. If breaking the build is not an option, one can choose monitoring instead, such as notifications to the developer or a monitoring dashboard. Another possibility is to allow build and deployment despite a certain (maximum) number of violations and reduce this threshold as errors are getting fixed. Yet another option is small and variable-size rulesets, where rules are introduced one by one, so that when all issues related to a given rule are fixed, a new rule comes into play.

Regarding other constraints, it was desirable to notify the developers upon new accessibility issues as soon as possible with tests that work on page fragments. This makes it easier to avoid introducing errors in the first place, and it makes teams less hesitant towards further integration because violations can be addressed before the centralized building process, which in itself increases testing complexity and the chances to break things. Therefore, implementing checkers locally for each developer seemed to be a

good solution, lowering the frequency of centralized build breaks, which in turn allows for larger rulesets or a lower error threshold.

For meaningful accessibility testing, there also is the need for integration tests that check entire pages, i.e., where each page element i validated in context. Such tests, however, cannot be invoked before any new code is fully integrated with the rest of the website, which often happens after the developer has shifted focus to other tasks. Thus, this kind of testing works better for plain monitoring and statistics.

Finn thus decided to employ multiple solutions. After a thorough assessment of available tools and rulesets, which are discussed in another article (Bai et al., 2019), selected Finn teams chose axe to test microservices/code fragments. The WCAG validator axe (Deque Systems, Inc., 2018) is run in the Javascript testing framework Jest (Various contributors, n.d.-b) and combined with the end-to-end testing tool Cypress (Various contributors, n.d.-a). Before the developers check in their code, they have to run a number of given tests for that microservice either locally or when the code is deployed in the software builder Travis. The axe tool will break the build if accessibility violations are found, and the developer will get the chance to commit fixes without running the entire process with deployment over again. However, as discussed earlier, Finn also decided to accept a certain number of issues in order to avoid a complete deployment stop, so it is possible to deploy code with accessibility issues. At the time of writing, axe is applied to the frontpage of the "Travel" section, the "Oddjobs" (Småjobber) module, and to Finn's frontend library, which contains the CSS code and about 20 smaller React modules like date pickers, form validation, and modal dialogs.

To test all modules in context of the entire page, the Finn teams use Pa11y and Pa11y Dashboard (Various contributors, 2018), which allows to validate multiple URLs. The dashboard displays the number of violations for each tested page and also any progress over time. It is shown on a big screen near the coffee machine of Finn's offices and is thus easily visible so everyone can see the status of some selected pages at any time.

4 SURVEYS

In order to evaluate changes to universal-design / accessibility aspects in the teams before and after the integration, two surveys were sent out to Finn's development teams; one before the implementation, and

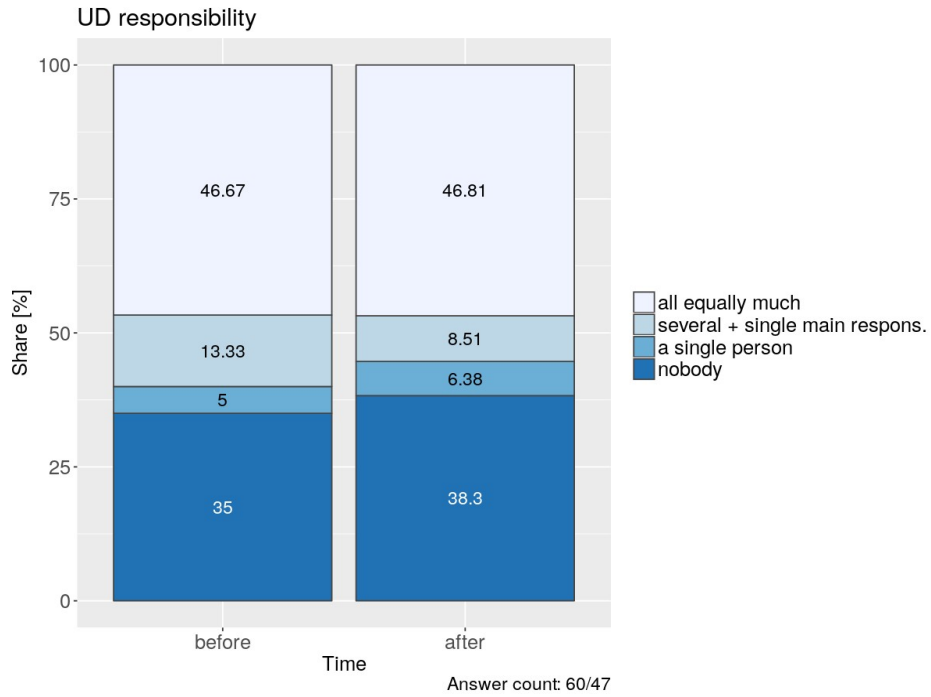

Figure 1. Answers regarding how the responsibility for UD is organized in the teams

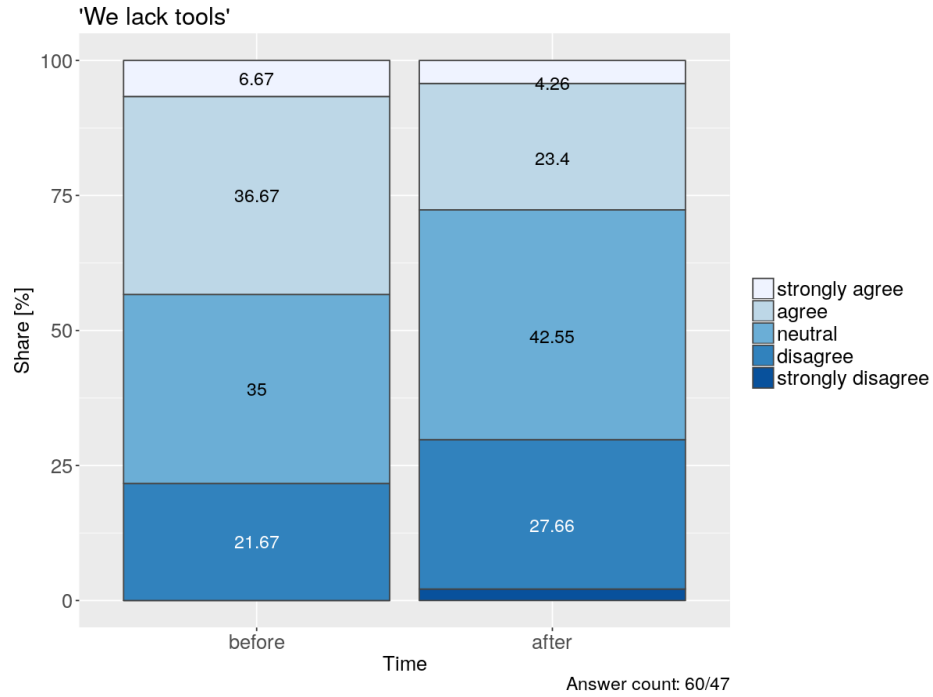

Figure 2. Answers regarding whether there is a lack of tools

one after. There were roughly two months in between both surveys. The surveys were sent out to entire teams and answered by 70 and 52 respondents, respectively, which indicates that the groups of respondents for both surveys were not entirely identical. The teams consisted of mostly developers (> 80%), team leaders (~10%), and system administrators / devops engineers.

Roughly 90% of the respondents reported to roll updates at least once per week or even daily (60-70%). More than 60% of the respondents rated the quality of the code they produce is high, and close to 40% said it is at least "OK". About 40% of the respondents commented that they use "a lot" of time on testing,

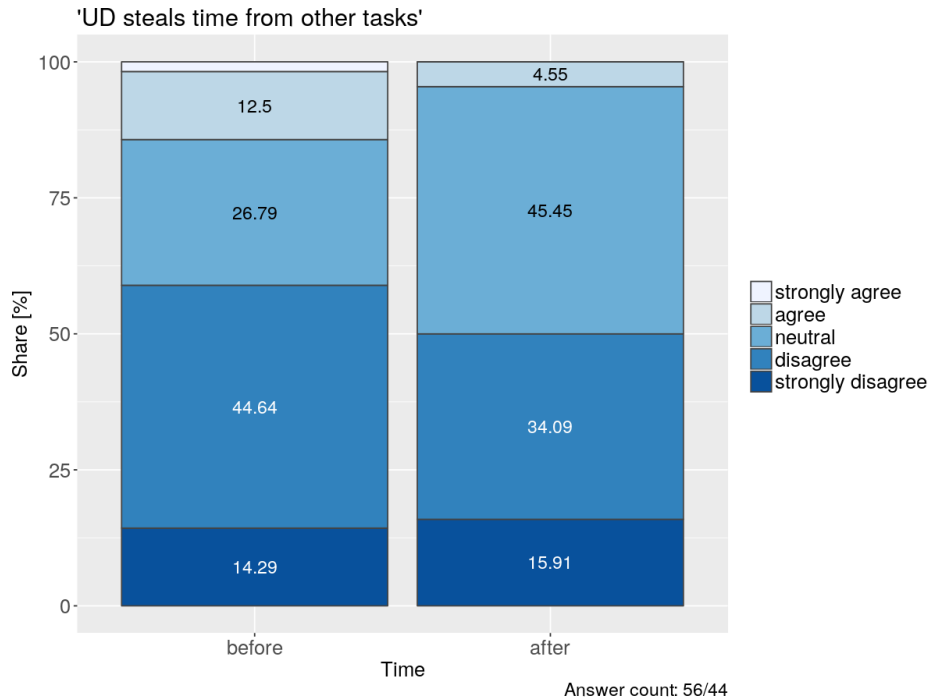

Figure 3. Answers regarding whether UD takes away time that should be spent on other tasks

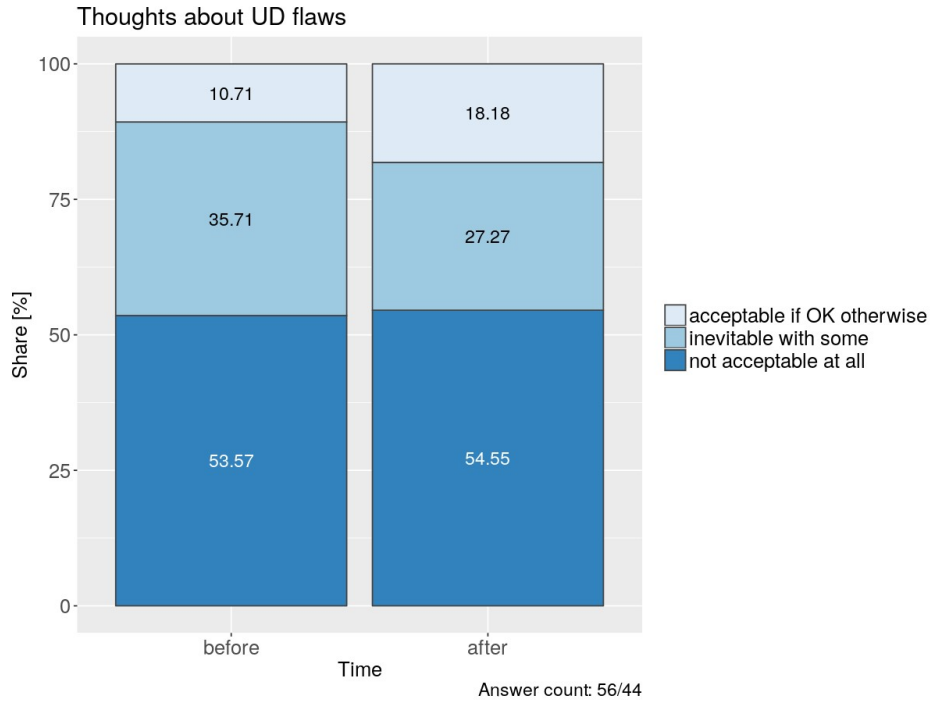

Figure 4. Answers regarding what the informants think about UD violations

while around 60% are neutral to this statement or even disagree, so we conclude that testing is not of the teams' highest concerns. This is also mirrored by the answers to the question about how well testing routines are anchored in the teams, where roughly half of the respondents stated they have very few or none routines at all (~10%) or only some, loosely integrated (~40%).

When it comes to universal design (UD), almost 40% of the teams report to have nobody to deal with those concerns, while ca. 6% say there is a single dedicated person, ~10% say there are several (one with the main responsibility), and nearly 47% of the respondents think that this is part of everybody's

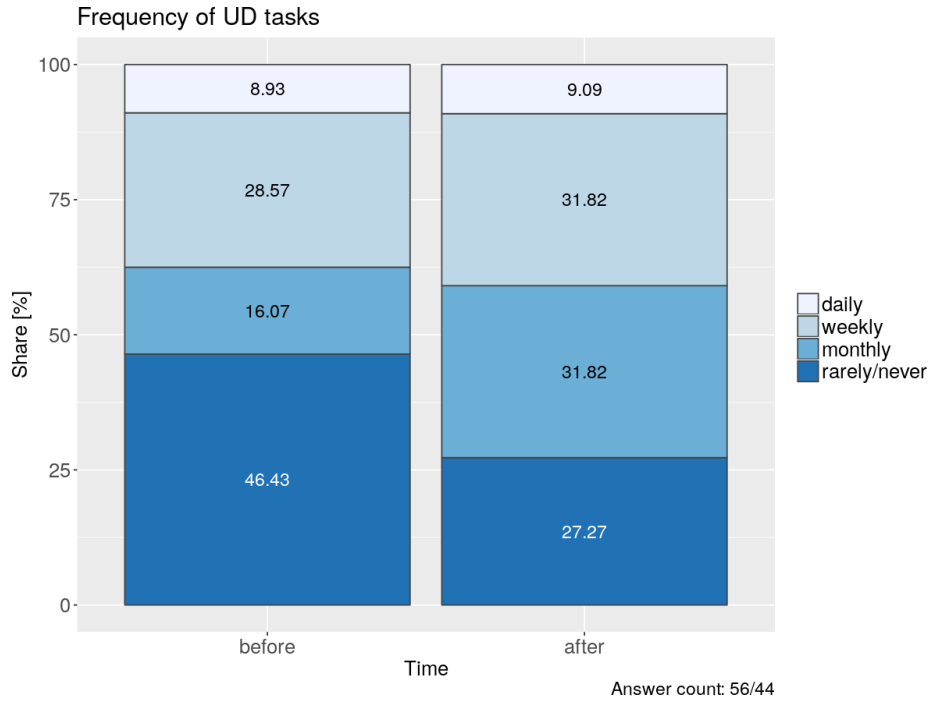

Figure 5. Answers regarding how often the informant works with UD

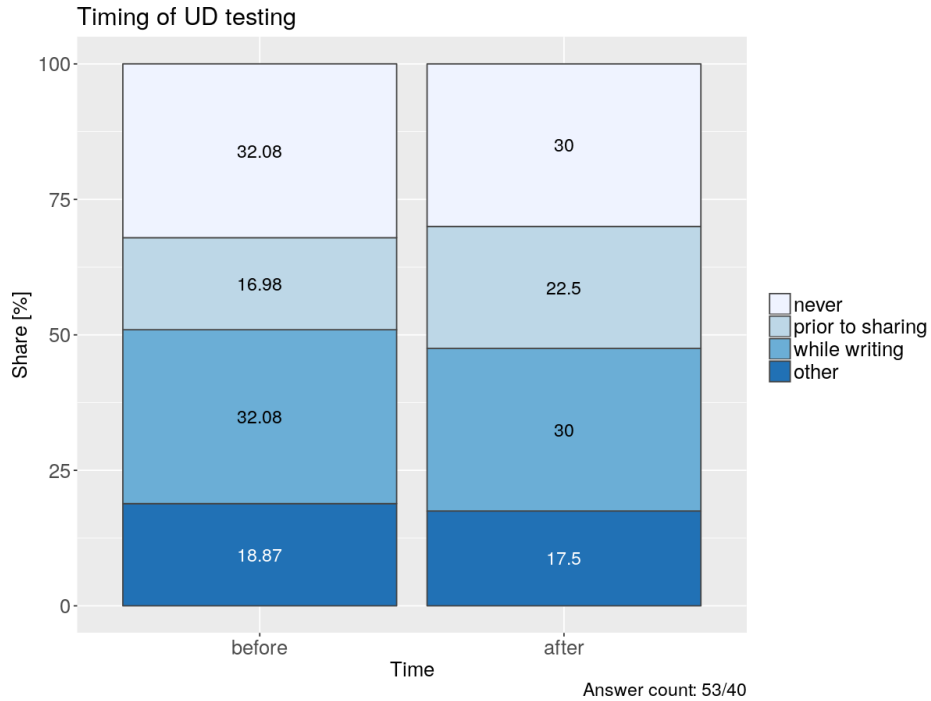

Figure 6. Answers regarding when UD testing is employed

responsibility, which is a rather high number, both on a national and international level. Before the integration, many respondents said their teams were in the need for an automated accessibility testing tool (~43% agree or strongly agree), while 35% were neutral, i.e., at least not refusing it. After the integration, the share of advocates for a tool had not surprisingly diminished (~27% agree or strongly agree), whereas the share of neutral answers had increased to almost 43%. This shows that the tools fulfill its purpose, but also that many still have a wait-and-see attitude, as the time for the teams to get acquainted with the solution was rather short (a few weeks).

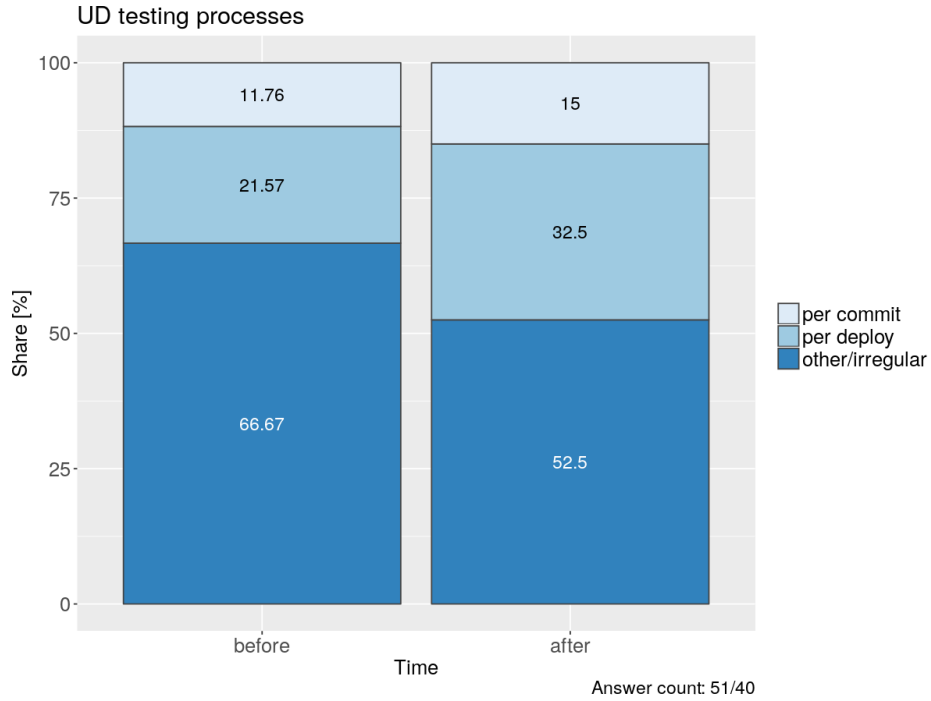

Figure 7. Answers regarding how often UD testing is employed

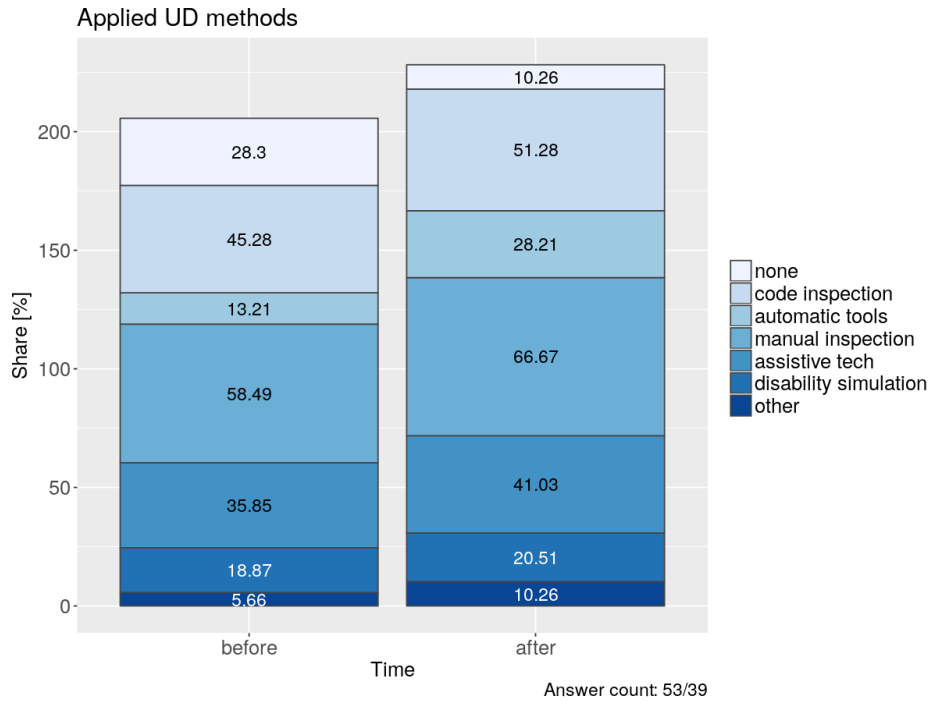

Figure 8. Answers regarding what methods are used by the teams to test UD aspects

Another question targeting the same topic showed a similar result, namely if the respondents think that UD takes away time from tasks they would prefer to do instead. Here, the share of those who were neutral to this statement increased considerably with the tool integration, from 27% to 45%. Simultaneously, the number of those who disagree/strongly disagree decreased somewhat from around 59% to 50%, most likely because more developers are now forced to address accessibility-related matters. On the upside, the share of respondents with a negative accessibility attitude (who agree/strongly agree with the aforementioned statement) diminished from ~14% to 5%.

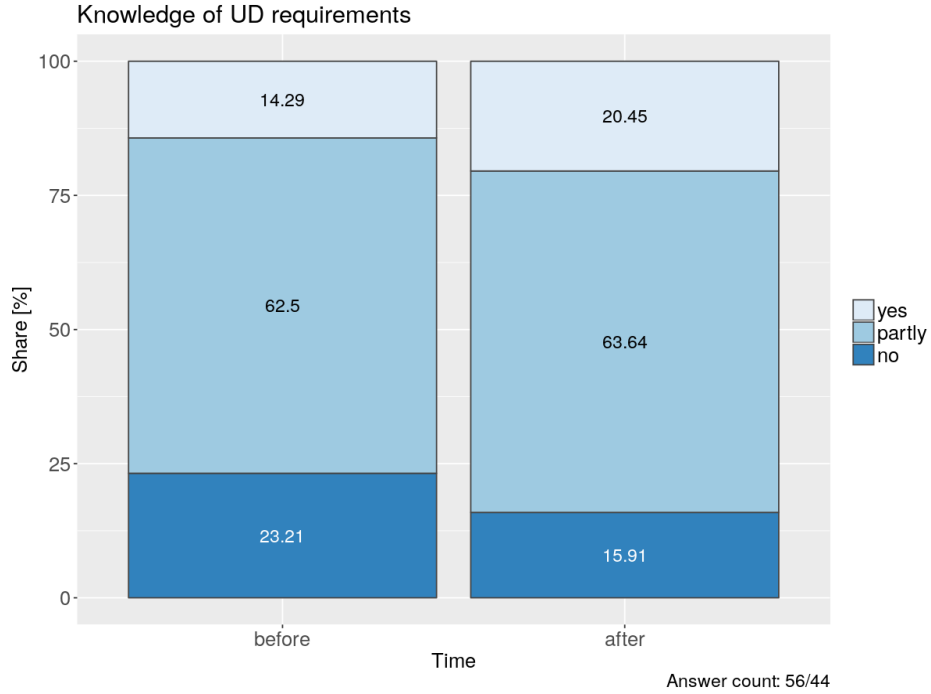

Figure 9. Answers regarding whether the informant knows the UD requirements well

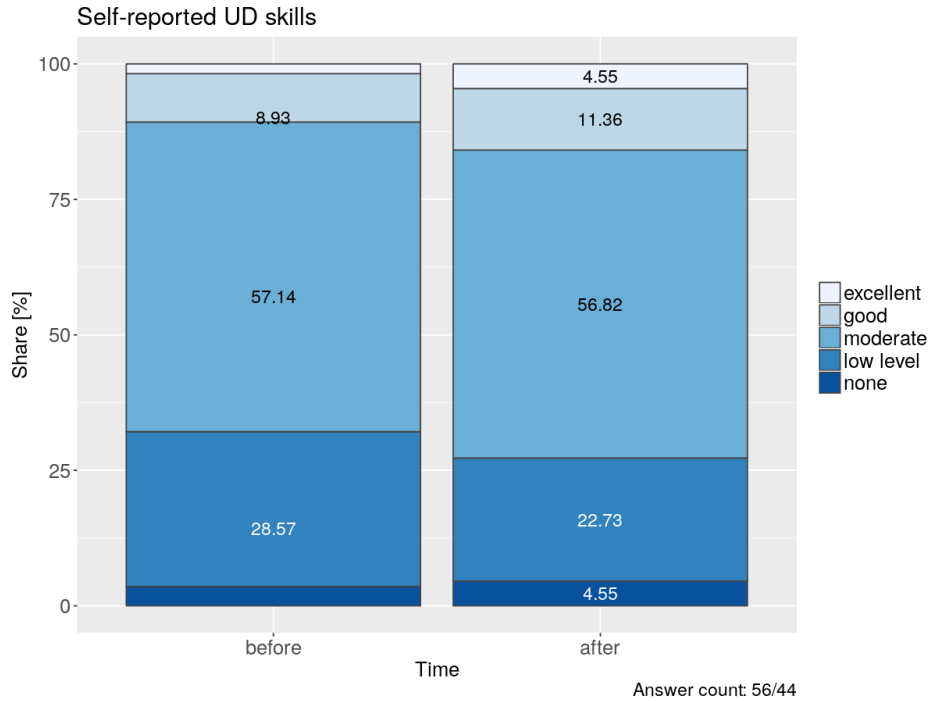

Figure 10. Answers regarding what is the self-rating of UD skills

The question regarding what respondents thought about UD/A11Y violations both before and after the integration lead to interesting answers. The share of those who thought A11Y flaws are unacceptable was almost constant (54% before and 55% after), which we believe can be accounted to the "hard core" of A11Y advocates; a pretty good result for itself. Why there were more after the integration (18% vs. 11%) who thought that A11Y flaws are acceptable if everything else is good enough, is not known, and our best explanation is that the groups of respondents differed slightly from survey to survey.

The integration affects mostly those respondents who rarely deal with A11Y-related tasks: The share of respondents who never/rarely do A11Y tasks was nearly halfed (46% before and 27% after integration), and respectively the share of those who do A11Y tasks less than each week almost doubled correspondingly (16% before and 32% after). The other shares, those who conduct A11Y work daily or on a weekly basis, were almost constant (9% and 29-32%, respectively). As can be seen in the respective graph, there is a slight (2%) reduction of respondents who never test, which unfortunately is not significant, but there is also an almost 6% increase of those who test for A11Y conformance before they share their code with others. The share of respondents who test for A11Y on commit or deploy supports this impression as it goes up from ~33% to ~45%, while those who test on an irregular basis or never goes significantly down from roughly 67% to 53%.

It can be concluded that the checkers in fact result in the positive impact they were designed to have. As seen in the corresponding graph, the introduction of the tools leads to increases of all kinds of A11Y testing, ranging from code inspection and the use of automatic checkers to testing by means of assistive technology and A11Y simulation, as well as other methods. A nice side effect of the A11Y tool introduction was that people were trained/educated in accessibility-related matters. During the second survey, more respondents claimed to know the UD requirements at least to some degree (roughly 84% vs. 77% before), but it has to be kept in mind that those self-reported numbers have to be treated with care. The answers correspond to those concerning the question how the respondents would rate their own UD skills on a Likert scale from one to five. While the share of the moderately skilled is approximately constant (~57%), the share of those with (self-reported) good and expert skills goes up from 11% before the integration to 16% after.

5 LIMITATIONS

This work is limited by the following considerations.

First of all, the meaning of self-reported data has always to be treated with caution, as the answers might diverse from objective measurements of behavior. Also, the two months in between surveys were due to a tight project schedule, and ideally there should have been more time before the second survey for the teams to get acquainted with the newly integrated tools. Next, having non-identical populations answering each survey is suboptimal and reduces the validity of answers somewhat. It was, however, not within our possibilities to control that all participants of the first survey, and only them, also would answer the second survey. In addition, we argue that multiple answering of a given number of questions is likely to lead to slightly different results for each participants each time anyhow, so one is not guaranteed the same results even though the population is identical. Regarding the size of the population, we think having 52 respondents during the second survey is satisfactory from a statistical point of view, but it would have been better to have approximately the same size as in the first survey. Finally, we would like to mention that the nature of such a survey makes a study like ours to lead to a somewhat superficial overview and does not allow to go in-depth on particular questions. For example, it has not been possible to answer *why* there were more after the integration who thought that A11Y flaws were acceptable.

6 CONCLUSION & OUTLOOK

In this work, we have discussed in how far the integration of automated accessibility checkers in a continuous deployment process can increase a company's awareness for universal design. The project's industry partner Finn successfully integrated the tools axe and Pa11y for respectively code and page fragments, and end-to-end testing. While the former was made possible by allowing a certain number of accessibility violations to avoid build breaks, the latter is employed for monitoring purposes only.

Two surveys, which were carried out before and after the integration of the tools, show that both tools have the anticipated impact: The number of persons that have to work with accessibility went up, and so did the time they have to spend on accessibility. Also, the teams' awareness for accessibility and universal design increased, and the same happened with the level of relevant skills in the various teams.

Regarding future work at Finn, new tests are going to be written for new modules when it is deemed practical to do so, but it is unlikely that there will be full accessibility coverage for all microservices in the near future. Also, there are unsolved challenges: One is microservices that demand login. Another is that a number of front-end services depend on other microservices, of which only a few have tests, so there is the risk that a particular service will break due to violations in some other service. While these problems are solvable, it will take time to get there. And finally, the dashboard also need some adjustments to make the displayed information more appealing and update more frequently.

References

Accessibility tools audit results - Overview - GDS accessibility team. (n.d.). Retrieved April 2, 2019, from https://alphagov.github.io/accessibility-tool-audit/

Bai, A., Fuglerud, K., Skjerve, R. A., & Halbach, T. (2018). Categorization and Comparison of Accessibility Testing Methods for Software Development. Studies in Health Technology and Informatics, 256, 821–831.

Bai, A., Skjerve, R., Halbach, T., & Fuglerud, K.S. (2019). Integrating accessibility testing in automated software build processes. Submitted to NIK-2019, Narvik (Norway)

Brajnik, G. (2004). Comparing accessibility evaluation tools: a method for tool effectiveness. Universal Access in the Information Society, 3(3-4), 252–263.

Centeno, V. L., Kloos, C. D., Fisteus, J. A., & Álvarez, L. Á. (2006). Web accessibility evaluation tools: A survey and some improvements. Electronic Notes in Theoretical Computer Science, 157(2), 87–100.

Deque Systems, Inc. (2018). axe: Accessibility for Development Teams. Retrieved from https://www.deque.com/axe/

DIRECTIVE (EU) 2016/2102 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 26 October 2016 on the accessibility of the websites and mobile applications of public sector bodies. , (2016).

European Commission. (2012, December 3). Digital Agenda: Commission proposes rules to make government websites accessible for all. Retrieved from http://europa.eu/rapid/press-release_IP-12-1305_en.htm

European Internet Inclusion Initiative. (2015, September). Checked sites from European websites. Retrieved March 27, 2018, from European Internet Inclusion Initiative website: http://checkers.eiii.eu/en/benchmarking/testrunresults/a6bc0b1d-598d-4c00-af2c-a0a073124c64

Fuglerud, K. S. (2014). Inclusive design of ICT: The challenge of diversity (p. 282). https://doi.org/10.13140/2.1.4471.5844

Halbach, T., & Fuglerud, K. S. (2016). On Assessing the Costs and Benefits of Universal Design of ICT. In H. Petrie, J. Darzentas, T. Walsh, D. Swallow, L. Sandoval, A. Lewis, & C. Power (Eds.), Universal Design 2016: Learning from the Past, Designing for the Future. Retrieved from http://ebooks.iospress.nl/ISBN/978-1-61499-684-2

Halbach, T., & Lyszkiewicz, W. (2015). Accessibility Checkers for the Web: How Reliable are they, actually? Proceedings of 14th International Conference on WWW/Internet (ICWI). Maynooth (Ireland): International Association for the Development of the Information Society.

Leitner, M.-L., Strauss, C., & Stummer, C. (2016). Web accessibility implementation in private sector organizations: motivations and business impact. Universal Access in the Information Society, 15(2), 249–260.

Namiot, D., & Sneps-Sneppe, M. (2014). On Micro-services Architecture. International Journal of Open Information Technologies, 2(9), 24–27.

Norwegian Agency for Public Management and eGovernment. (2018, December 13). Statusmåling 2018 - Oppsummering | Universell utforming. Retrieved March 27, 2019, from https://uu.difi.no/tilsyn/statistikk-og-undersokelser/statusmaling-2018-oppsummering

Parnin, C., Helms, E., Atlee, C., Boughton, H., Ghattas, M., Glover, A., … Williams, L. (2017). The Top 10 Adages in Continuous Deployment. IEEE Software, 34(3), 86–95.

Petrie, H., King, N., Velasco, C., Gappa, H., & Nordbrock, G. (2007). The usability of accessibility evaluation tools. In Universal Access in Human-Computer Interaction. Applications and Services (pp. 124–132). Springer.

Proposal for a DIRECTIVE OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL on the approximation of the laws, regulations and administrative provisions of the Member States as regards the accessibility requirements for products and services. , (2015).

Putnam, C., Dahman, M., Rose, E., Cheng, J., & Bradford, G. (2016). Best Practices for Teaching Accessibility in University Classrooms: Cultivating Awareness, Understanding, and Appreciation for Diverse Users. ACM Transactions on Accessible Computing, 8(4), 13:1–13:26.

Regulation for universal design of information and communication technology (ICT) solutions. , (2013).

Røssvoll, T. H., & Fuglerud, K. S. (2013). Best Practice for Efficient Development of Inclusive ICT. In C. Stephanidis & M. Antona (Eds.), Universal Access in Human-Computer Interaction. Design Methods, Tools, and Interaction Techniques for eInclusion (pp. 97–106). Springer Berlin Heidelberg.

Tanaka, E. H., & Da Rocha, H. V. (2011). Evaluation of web accessibility tools. Proceedings of the 10th Brazilian Symposium on Human Factors in Computing Systems and the 5th Latin American Conference on Human-Computer Interaction, 272–279. Brazilian Computer Society.

Various contributors. (2018). Pa11y. Retrieved from http://pa11y.org/

Various contributors. (n.d.-a). JavaScript End to End Testing Framework | Cypress.io. Retrieved April 2, 2019, from Cypress.io website: https://www.cypress.io/

Various contributors. (n.d.-b). Jest · Delightful JavaScript Testing. Retrieved April 2, 2019, from Jestjs.io website: https://jestjs.io/

Vigo, M., Brown, J., & Conway, V. (2013). Benchmarking Web Accessibility Evaluation Tools: Measuring the Harm of Sole Reliance on Automated Tests. Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility, 1:1–1:10. New York, NY, USA: ACM.
Watanabe, W. M., Fortes, R. P. M., & Dias, A. L. (2012). Using Acceptance Tests to Validate Accessibility Requirements in RIA. Proceedings of the International Cross-Disciplinary Conference on Web Accessibility, 15:1–15:10. New York, NY, USA: ACM.