**Software Engineering for Accessible Computing: How Accessibility Affects the Software Development Life-Cycle**

By

Brian Worts

CSC 415: Software Engineering

Date of Submission: 11/7/2019

# **ABSTRACT**

Accessible design principles are extremely important to any software given the wide range of users they may effect. Primarily, these users are members of disenfranchised or groups discriminated against that are protected under the law. A software engineer must possess knowledge of these design principles in order to be an effective engineer. They need a certain level of knowledge regarding accessible computing, including both the ethical and legal aspects. They must also be aware of the various effects that accessible computing may have on software and software development processes.

**Keywords**

Accessible computing, inclusive software, human factors, universal design

1. **INTRODUCTION**

The Americans with Disabilities Act (ADA) prohibits discrimination against individuals with disabilities in all areas of public life. It guarantees equal opportunities and equal rights for people, regardless of any disability they possess [Americans With Disabilities National Network]. This act did a lot of great things for lowering barriers to help people with disabilities live the same life as people without them, and is not the only one of its kind, but these acts also created new challenges for people like software developers. For example, software applications, like word processors or web browsers are vital to a productive work environment, and need to be accessible to everyone, no matter the disability under the ADA. Similarly, websites need to be designed with this in mind as well, since poor design may cause an extra barrier for people with disabilities that prevents them from completing a task, such as a person who is dyslexic being unable to read the fonts used, resulting in ADA non-compliance [U.S. Department of Education]. Thus, software developers need to keep “accessible computing” in mind.

“Accessible computing” refers to the process of making software that is just as accessible to those who have disabilities as to those without. Not only is this a legal obligation for developers, but also a moral obligation that has ramifications on all aspects of software design. There will be more hurdles when it comes to developing the functionality of the product, more time spent on managing the project, more money needing to be allocated to various branches of the project, and significantly more effort put into testing and reviewing the product.

**2.0 IMPACT ON SOFTWARE REQUIREMENTS**

**2.1 Determining Accessibility Requirements**

When determining accessibility requirements, developing for the minimum accessibility requirements can result in the requirements for inclusive software being met, but the product being very difficult or sometimes impossible to use. [Newell A. F. 2008]. In other words, accessible software should be designed with the individual in mind, not just the requirements. A designer may successfully allow a user to navigate to all pages via keyboard shortcuts, but the order of the pages may be completely illogical or there may be a ridiculous amount of steps to get somewhere, as compared to a user that can find the link they are looking for in one click. As with all use cases, a designer must think of the “human factors” that make each user unique and design programs with those people in mind. This may seem daunting, but with the proper planning and knowledge it is a very reasonable task.

## **2.2** **Combining Functionality and Accessibility Requirements**

A good developer will incorporate functionality related to accessibility with all other general functional requirements. By developing accessibility functionalities in tandem with the other functionalities, not only is it less likely to reduce waste of resources due to changing finished functionality, but can also result in a better experience for all users [Newell A. F. 2008]. In some cases, such as with subtitles in a video player, functionality developed to make a product more accessible gives the average user an alternative way to use the product incase their preferences for how they watch a video change depending on their situation. Another example could be keyboard shortcuts in a program that allow a user to navigate the pages. A user working on something, whose mouse becomes non-functional or maybe has something else in their hand, can make use of these keyboard shortcuts developed for accessibility as well. Thus, by thinking of accessibility not as separate, but as part of the functionality of a project, it can result in a better overall product for all users.

**2.3 Compromising to Achieve Accessibility**

In some cases, designing a product that is usable by everyone may result in a developer resorting to making bad compromises in terms of the functionality or design in order to reach accessibility requirements. Compromising, just to make a product more accessible, is a bad programming practice and may result in unhappy customers that are not getting what they think they paid for [Zimmerman, Gottfried 2007]. It is important that a developer keeps this in mind, since failing to deliver on something like functionality can have devastating effects on a project. Compromising may also result in wasted development resources since cutting or “dumbing down” functionality means that effort was put in to something that did not end up in the final product. An example of compromises that may hurt the overall product are using color schemes or layouts that may lower the aesthetic appeal of a product. Other examples could include things like removing the ability to scroll or zoom since they could make software navigating a barrier to the disabled. As a result, not planning properly for accessibility can have serious ramifications on a project.

**2.4 ADA on the Internet**

The ADA applies not only to the workplace, but also to websites as well. This is because under the ADA, websites are considered a public space [U.S. Department of Education]. Thus, people who get protection under the ADA are protected on the internet just as much as they are in a restaurant for example. Plenty of well-known companies have had similar things to this hypothetical example happen to them as well. Bank of America, H&R Block, CVS, Staples, Target, and many more companies have lost lawsuits over their failures to have ADA compliant websites, primarily as a result of discriminating against the blind and visually impaired [Newell A. F. 2008]. Therefore, accessible computing affects not only the software being developed, but can also have huge ramifications on the stakeholders as well.

**3. IMPACT ON PROJECT MANAGEMENT**

**3.1 Project Planning**

Any good software project will devote considerable time to the planning stages, and projects that do not may find their developers, products, or customers suffering as a result. During the planning stages, designers who are keeping accessible computing in mind will find themselves tasked with planning not just the major functionalities and design of the project, but also on different ways to access that functionality, like determining how a blind person may work with the product. Often, this results in designers beginning to feel extremely burdened with the idea that their product must work for “everyone”, along with the additional use cases that now must be considered. This can result in a lack of effort and frustration from developers. Typically whenever this happens, projects suffer [Burgstahler, Sheryl 2014]. Thus proper planning must be done to allow for correct allocation of resources and time in order to keep developers from feeling burdened with the weight of accessible design.

Failing to properly plan for the accessibility of a product may also result in the functionality not being considered or implemented until all other functionality has been completed. This makes accessibility more of an afterthought, patronizes those which need these features, and may have destructive effects on a product [Burgstahler, Sheryl. 2014]. Accessible computing is not just a legal requirement, but also a moral obligation for developers. Tacking accessibility on at the end is likely to make the features not effective for those that may need them and therefore useless, so resources may be better spent elsewhere.

Not planning for accessibility throughout the project and instead trying to add them on at the end, can also greatly harm a project. For example, after most functionality is completed, a file type may need to be changed from a PDF format to a format that is more text to audio friendly to help those that are visually impaired. Not considering this until the end may also result in completely breaking the functionality of a project, such as if there are other files that were dependent on that PDF file that was changed. As a result of this, proper planning for accessible computing must be taken into account early in the development stage or else a product or user base may suffer.

Despite efforts being made towards advancing accessible computing, there are still many software developers that are unaware of its importance. One such study of twenty-five companies that produce educational software in 2001 found that only 10% were aware of accessibility issues. In addition, none of these companies were addressing accessibility in product development and most importantly, 88% had no plans to address these accessibility in the future [University of Washington]. Although this survey is somewhat outdated, its results are still very relevant today as companies as large as Morgan Stanley have been sued as recently as 2018 for having an inaccessible website to the blind [Brockman, Joshua. 2019]. Given how website accessibility has been an issue since the beginning of the internet, it shows how largely this problem is ignored by the general public.

**3.2** **Budgets and Time**

While it is possible that budget and time constraints are not a developer’s number one priority, they are often the thing at the front of management’s mind. An inexperienced project manager may not realize the importance of something like implementing accessibility, so in their short-sightedness, they decide they will implement it at the end or prioritize other functionality. One such study found that costs can be up to ten times as much for website design if accessibility is added on at the end as opposed to added the same time as the other functionality [Zimmerman, Gottfried. 2007]. As a result of studies like these, many wrongly view accessibility as an unaffordable addition to a project. In reality, making accessible software is a very realistic and relatively cheap process, so long as proper planning and resource allocation is done.

## **3.3** **Software Review Phases**

Accessible computing introduces many new variables into the review phases of software. As a result of the increase in range of users, developers may find themselves needing to be extremely creative in developing test cases to cover the range of users they are developing for. Luckily, entities like the United States Justice Department offers a comprehensive list of items they look for in deeming a software to be accessible [U.S. Department of Justice]. This list can be used in the testing phase as general items that, if met, will cover most test cases related to accessibility. It includes about thirty questions such as:

* Does the software provide keyboard equivalents for all mouse actions, including buttons, scroll windows, text entry fields, and pop-up windows?
* Does the program provide clear and precise instructions for use of all keyboard functions as part of the user documentation?
* If information is provided in an audio format, is it also capable of being displayed by the user in a visual format?
* Does every window, object, and control have a clearly named label?
* Can a user adjust or disable flashing, rotating, or moving displays?

Although this list should not be used exclusively, it serves as a good base-line for developing tests to make sure software is accessible. It is significantly easier to develop test cases when using a list like this as opposed to trying to come up with test cases by oneself. Even if in contact with a disabled person, it is important to keep these guidelines in mind since even they might not be aware of what exactly a software engineer’s legal requirements are.

Additionally, testing will need to be performed using assistive technology. Assistive technology is any hardware or software used by a person to help overcome their disability . This can include things like wheelchairs, special keyboards that allow for easier input, or screen readers for the visually impaired that translate text to speech[University of Washington]. As with all software products, testing needs to be done on not only the product, but also done using any hardware or software that may be utilized to access the product in the same way that a web developer should test their websites in as many web browsers as possible. Without proper testing, users that need assistive technology to use a computer may have extra barriers as a result of their assistive technology not functioning properly with the product. This would mean that the product is not ADA compliant and could result in repercussions on the developers.

**4. ACCESSIBILITY’S IMPACT ON TESTING**

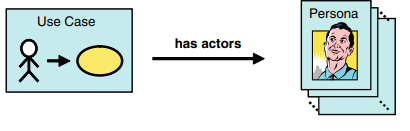
**4.1 Methods for Planning and Testing of Accessible Software**

The idea that a software needs to be accessible by everyone may cause a developer to become rightfully overwhelmed by the scope of such an idea, nor is it helpful when it comes to identifying stakeholders. In order to prevent such a thing from happening, there are design methods that counteract becoming overwhelmed by designing for “everyone.” One such method involves focusing on small sets of users first and identifying key characteristics that are relevant to the functionality of the product [Newell A. F. 2008]. This allows developers to identify “extreme users” which, when combined into one group, cover most or all of the functionality that will need to be added to make the product accessible.

# 4.1.1 A More Personalized Approach to Extreme Users

One problem that may result from making a group of “extreme users” is that even after finding this group, the accessibility functionality may still be too broad to implement all of it. In order to solve this issue, some research suggests a more personalized approach. While it is still helpful to find the extremes that will need to be developed for, a personal relationship between the developers and the members of the group can be even more beneficial. This may help developers gain a better idea for a person’s needs as well as empathy for the people they are developing. This allows a developer to see them more as “people” as opposed to seeing them as a “set of characteristics” [Newell A. F. 2008] that need to be able to access the product.

# 4.1.2 Using Actors

Another such approach for planning and testing is the idea of using “actors.” In some cases, it may be difficult or not within a projects constraints to spend time with people who are actually suffering from the disabilities being developed for. In order to get around this, developers may make use of actors, which are people that may play the role of a person with a disability while testing or planning a product [Zimmerman, Gottfried. 2007]. If these actors are one of the developers or a person with extensive software knowledge, this method has the added benefit of being more likely to test the boundary cases since the actors may know exactly where the boundaries are. A typical use case diagram with the addition of actors is shown in Figure 1. While this method is effective, nothing is more accurate than testing the software with the actual people that will be using it.

**Figure 1: A use case diagram that takes actors into account [Zimmerman, Gottfried]**

# **5. CONCLUSION**

Accessible computing is not just a moral requirement for developers, but also a legal requirement that affects every aspect of the software development life cycle. As explained, good programming practices have accessibility incorporated from the beginning of development. Accessibility requirements should be gathered at the same time as other functional and non-functional requirements are being gathered, so as to ensure they are being incorporated with other functionalities. Failure to do so can result in things such as failure to implement the accessibility functionality successfully thus leaving the company open to ADA complaints and lawsuits, and also can result in greatly increasing the development time and costs by up to ten times as explained in section 3.2. Significantly more time will also need to be dedicated to the testing and review phases of the projects. Testing will need to be done for the wide range of assistive technologies as well as the wide range of users that will use the product. Luckily, there are many ways of doing this testing including the use of “extreme users” and actors.

With all of this in mind, it is clear that accessible computing is something that should not be overlooked, but often is. It is much easier to forget about the needs of the disabled and develop programs for the majority of users, and unfortunately many choose this path. However, the last thing that these disabled individuals need are more barriers in their day to day lives. Though there are laws such as the ADA that protect these people from discrimination, it does not mean that they are free from it. Thus, it falls on people like developers to keep accessible computing in mind to make sure that those with disabilities can live with as few barriers to software access as possible.

# **6.** **Bibliography**

[1] Americans With Disabilities National Network. “What is the Americans with Disabilities Act ppppppp(ADA)?” <https://adata.org/learn-about-ada>. Retrieved November 2019.

[2] Brockman, Joshua. “At Banks and Fund Firms, Access Is Too Often Denied, Blind and Deaf Investors Say” The New York Times. July 5th, 2019. <https://www.nytimes.com/2019/07/> 05/ business/retirement-planning-disabled-deaf-blind.html.

[3] Burgstahler, Sheryl. 2014. “Designing Software that is Accessible to Individuals with ……….Disabilities”, University of Washington. <https://www.washington.edu/doit/sites/> ……….default/files/atoms/files/Designing-Software-Accessible- Individuals-Disabilites.pdf.

[4] Newell A. F. 2008. Accessible computing – Past trends and future suggestions. ACM Trans. ……….Access. Comput. 1, 2, Article 9 (October 2008). DOI : 10.1145/1408760.1408763. ……….http://doi.acm. org/10.1145/1408760.1408763.

[5] University of Washington. What are issues related to the design of accessible software? AccessComputing: The Alliance for Access to Computing Careers. <https://www.washington>. edu/accesscomputing/what-are-issues-related-design-accessible-software

[6] U.S. Department of Education. Statement on Requirements for Accessible Technology, February 2001. <https://www2.ed.gov/fund/contract/apply/clibrary/software.html>.

[7] U.S. Department of Justice. Software Accessibility Checklist. ppppppp<https://www.justice.gov/crt/software-accessibility-checklist>. Retrieved November 2019.

[8] Zimmerman, Gottfried. 2007. “Accessible design and testing in the application development

……….process: considerations for an integrated approach”. Springer-Verlag

……….DOI: 10.1007/s10209-007-0108-6.