“Move Fast and Break Things”: How Modern Software Development Practices Leave Some Users Behind

A Research Paper Submitted to the Department of Engineering and Society

Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science, School of Engineering

By

Nathanael Strawser

Spring, 2020

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

Advisor

Kathryn A. Neeley, Associate Professor of STS, Department of Engineering and Society

Introduction

"While software developers are directly concerned with programming, they contribute indirectly to profound changes in the working life of the users of their programs."

-- STEPS to Software Development with Users (Floyd, 1989)

In an era of fierce market competition, many companies work under tight deadlines and unveil software without fully understanding the needs of users and appreciating the role that this software will have in their daily lives. As Austrian computer scientist Christiane Floyd discussed in her writing on software development, understanding the role that users play is a critically important step in the process of crafting great software. Ultimately, this means software development is "a learning process for both developers and users" (Floyd, 1989, p. 5). Users adapt to new computer systems and changes in competence, while developers must understand how users interact with software.

As the world becomes more digitized, it is important to understand how users can be incorporated into the process of software development to increase usability rates and decrease user frustration. Specifically, the impact that multi-generational users has on the ultimate usability of software requires further understanding. However, research is mixed on the benefits of incorporating users into the software development process. While it may be necessary to fully understand nuanced needs of users (Kane, 2019, n.p), delaying the release of software to fully understand these challenges may be detrimental as well.

In this paper, I will seek to understand the challenges that users from different age groups face when using software applications and argue that the current practice of agile development methodology does not sufficiently advocate for the needs of all users.

Part I: Internet use by older adults is increasing, but usability is not

There are several known challenges that have been identified by older adult users when interacting with digital media, such as small font sizes, challenging interfaces, and a general sentiment of exclusion from online content. Sentiments that the online world was "created with someone very different than me in mind" are pervasive among senior citizens (Kane, 2019, n.p.). One user stated the following:

"You look at things that are on the internet and it's skewed towards not my demographic. The younger people, this is their medium. People my age did not grow up with it. People my age are not in charge of it."

Additionally, a large number of older adult users have one or more comorbidities that impact the way they use the internet. According to the Administration on Aging, 44.5% of Americans aged 65-69 have a chronic disability, and 73.6% of those 80 or older do. The intersectionality between age and health is one that should be factored in during software design in order to fully understand usability concerns from different groups. Some examples of these challenges include difficulty reading small fonts and error messages, the inability to click small buttons due to challenges with fine motor skills, and confusion when navigating on websites with multiple pages for those with poor memory recall.

Not all websites are created equal. Research by Becker found that commercial (.com) sites "were not senior-friendly in terms of providing help, contact, site maps, or privacy features on homepages," while "state government Web sites were more senior-friendly," likely due to more stringent usability requirements (2004, p. 400). As seen in Figure 1 below, 100% of commercial sites sampled had font sizes smaller than 12 point, which can be challenging to read.

Usability Issue	Problem	State (.gov)	Commercial (.com)	Nonprofit (.org)	Newspaper (.com)
Pull-down menus	Precise movement of the mouse may be physically challenging.	38%	12%	24%	8%
Small font size	Font size is difficult to read when it is smaller than 12 point.	84%	100%	96%	100%
Screen is 3 or more pages	Lengthy page requires memory recall of Web content.	10%	20%	4%	72%
No help feature	A question not supported by help feature may render the site inaccessible.	36%	52%	64%	48%
No contact us	No means of personal contact makes it difficult to obtain additional information.	14%	52%	4%	48%
No privacy statement	The user may mistrust the site when the use of personal information is unknown.	8%	36%	44%	40%
No site map	Complex sites may be difficult to navigate with no visual relationship among pages.	44%	88%	68%	56%

Figure 1. Guidelines for Making Senior-Friendly Websites (Becker, 2004, p. 389)

While the digital literacy of older adults has seen consistent evolution over time, evidenced by the increased use of ad blockers and modified search behaviors, these changes are not enough to offset the decline in overall usability as we age. Nielsen Norman Group (NNG), known for their research on user experience, indicate that people's ability to use websites between the ages of 25 and 60 declines by nearly 1 percent per year. Previous research by NNG includes one on one time with users and design sessions, but one area that appears to be left unexplored is an overview of which specific features of various applications that users in this age range find most useful.

The amount of adult internet users in the US has been steadily increasing and currently sits at an all-time high with 90% of adults online (PEW Research, 2019, n.p). Figure 2 below also indicates that 88% of adults 50 to 64 and 73% of adults 65 and older use the internet. The demand for quality software by older adults is only increasing, and it is unproductive and alienating to put these needs on the backburner.

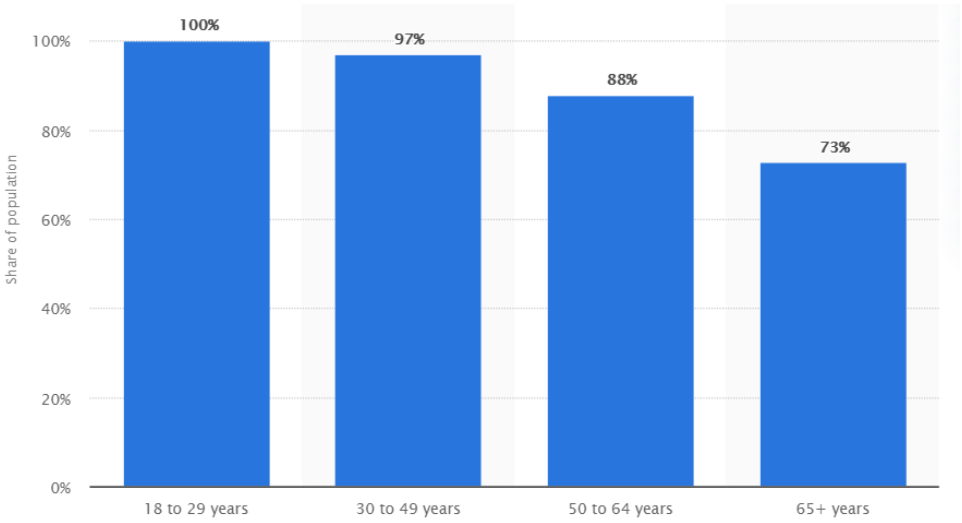

Figure 2 Internet Use by Age in the US. (Clement, 2019, n.p)

There are many known factors that improve software accessibility for older adult users and those with disabilities, published by the US National Institute on Aging and the US National Library of Medicine. This includes many front-end improvements such as using larger font sizes, reducing the use of moving interfaces and nested menus, crafting precise error messages, and providing hierarchical site maps (Morrell et al., 2019). The addition of hyperlinks with no information text (e.g. "Click here") makes navigation very difficult for those using text-to-speech software, and websites that do not scale properly make reading difficult for those who must browse at increased zoom. However, it is often not clear until after interacting with a website if it follows these standards.

Using software through assisted means often adds an additional burden and makes navigating complex and information-rich applications even more challenging. Redish (2004, n.p.) found that most users "do not use all the functionality of their software" and "do not know how to customize all the aspects they want to change." The focus should therefore be on maximizing the utility of the most commonly used features of software when it is being developed.

Due to the nuanced needs of this user group, it seems necessary to engage the users during the design of the application. However, user-centered design brings about many challenges, including slower development times and contradicting inputs. User-centered design is intended to "explicitly understand users and context of use by involving users throughout the design and development process" (Eshet & Bouwman, 2019, p. 1). Some studies have shown that user-oriented software design was directly related to "low overall success, few innovations, little flexibility, low team effectiveness, and low changeability of the software" (Heinbokel, 1996, p. 232). It also poses a challenge in the case where there do not exist "any prototypes or relevant systems" to present to users for feedback.

Part II: Agile methodology is a double-edged sword

When analyzing systems that must balance the needs of multiple stakeholders, it is helpful to think in terms of a framework such as actor network theory, initially outlined by Bruno Latour, among other scholars. Latour's "Where Are the Missing Masses? The Sociology of a Few Mundane Artifacts" provides a framework for weighing the benefit of various artifacts in everyday life. This framework aids in the analysis of creating, distributing, and maintaining software. There are several aspects to this. We can consider tests in code as a way to reduce the need for human actors via manual testing. Additionally, the process of creating software does not exist in a vacuum. Developers must balance the needs of the customers, users, regulators, and the economic interest of their employer.

Research by Eshet and Bouwman on user-centered design further discusses the challenges of modern software development and highlights many of the challenges of user-centered design. They indicated that software projects "are complex social systems that are embedded in a dynamic, often inter-organizational context" (2019, p. 1). Large projects

include users with very diverse views, and attempting to cater towards a subset of these poses many challenges. Furthermore, users and customers of software are not universally one and the same. While some software may be contracted for a specific team, it may also be bought by customers who may or may not use the software themselves (e.g. an executive member purchasing software to be used by employees at a firm with locations across the country). This presents a large barrier to meaningful user interaction.

The traditional software development lifecycle (SDLC) includes gathering requirements, design, development, testing, deployment, and maintenance. In the past two decades, there has been a large shift in the culture of software development towards agile methodology in an effort to remain competitive. Goldman (1995) states the following about agility as a business concept:

"Agility is dynamic, context-specific, aggressively change-embracing, and growth-oriented. It is not about improving efficiency, cutting costs, or battening down the business hatches to ride out fearsome competitive 'storms.' It is about succeeding and about winning: about succeeding in emerging competitive arenas, and about winning profits, market share, and customers in the very center of the competitive storms many companies now fear."

Applied to software development, agile methodology "is the use of light-but-sufficient rules of project behavior and the use of human- and communication-oriented rules" in order to scale quickly and reach the market faster (Cockburn, 2009).

One of the most influential works surrounding the process of software development is "The Agile Manifesto," which highlights twelve key principles that aim to create better software projects. Two of these principles, however, pose challenges for groups working on large-scale projects or on projects a wide array of users. These principles are 1) "Agile

processes promote sustainable development. The sponsors, developers and users should be able to maintain a constant pace" and 2) "The most efficient and effective method of conveying information to and within a development team is face-to-face conversation."

"The Agile Manifesto" highlights five key factors that can be used to determine whether a specific software project should take on a more agile or plan-driven framework. While agile methodology thrives best with "dedicated onsite customers, focused on prioritized increments" there is little discussion about the actual feasibility of such a requirement. Additionally, this distinction is lacking from the polar decision chart shown in Figure 3 below. Instead, the factors are focused on the dynamics of the software team itself (i.e. personnel, dynamism, culture, and size) and the nature of the project (criticality).

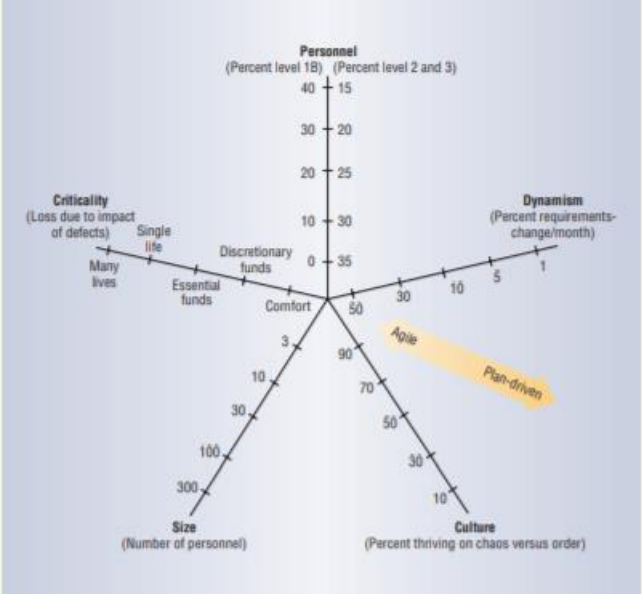

Figure 3. Agile polar chart. (Boehm, B., & Turner, 2003, p. 59)

The consequences of a rushed software development process are best analyzed through a case study. On February 3rd, 2020 Iowa held their Democratic caucus. This year, they used a new mobile application to tally the results. However, a technical issue

prevented the app from tallying results correctly – instead, it froze when users tried to do so, but the challenges do not stop there. When users tried to call the help desk, they found a single employee who "did not always respond to calls and emails" (Warzel, 2020, n.p.). Additionally, when volunteers tried to log into to a secure system to tally the results as they were called in, they found they needed a smartphone to retrieve a code, which they were explicitly told not to bring. This highlights the different interests of various stakeholders of the software. These include the software developers, help desk personnel, the users (Democratic party volunteers), and organizational-level staff. One clear takeaway from this debacle is that software cannot exist in a vacuum. Simply introducing a new application with little forethought can pose many issues. In this case, the technical issues were overshadowed by the larger organizational missteps.

Another lesson from this interaction is how engineering can serve as social experimentation. In their work on this topic, Martin and Schinzinger state that "as the ultimate test of a product's efficiency, safety, cost-effectiveness, environmental impact, and aesthetic value lies in how well that product functions within society, monitoring cannot be restricted to the in-house development or testing phases of an engineering venture. It also extends to the stage of client use. Just as in experimentation, both the intermediate and final results of an engineering project deserve analysis if the correct lessons are to be learned from it" (Martin, M. W., & Schinzinger, 2010, p. 79). In the case of the Iowa caucus, the lack of a full system test and reliance on the users to figure out bugs in the system demonstrates how the software development process can rely too heavily on the users themselves to do the heavy lifting. The current agile software development process is partially to blame for this; unrealistic deadlines and the prioritization of developing

feature-full applications quickly without a full understanding of user needs increases the probability of failure.

Even when developers thoroughly test software in-house before rolling it out to users, homogeneous demographics within a company may mean that test results are not comprehensive. Google, for example, reported in 2019 that 7.5% of its employees have a disability, nearly half the national rate. Thus, while in-house testing may seem like a possible solution given the difficulty of accurate user representation in the software development process, even this method poses issues if the testing is conducted with teams that have dissimilar demographics from their intended user base, including older adult users and those with disabilities.

Part III: Software developers must synthesize multiple perspectives during the development process

Analysis of current software development patterns, user-centered design, and current usability standards helps us answer the question of how users can be integrated into the software development process to increase usability across age groups.

As discussed in Part II, agile development's focus is on optimizing for business needs and maximizing revenue. This process has advantages, such as the ability to quickly pivot if needed since development is incremental. However, I argue that agile methodology may be detrimental to the needs of end users. Since much of the focus is on rapid iteration and business potential, nuanced user concerns may be pushed to the side. The Iowa Caucus app discussed previously may be one such example of this: the agile development process places pressure to iterate quickly and beat competitors to market at the cost of thorough testing and a deep understanding of the user base.

Furthermore, as the software development process is becoming more agile in nature, the users of software are also shifting over the years. These two forces may in some ways oppose each other. Shortcuts taken during the agile process that result in inefficient software, challenging user interfaces, and critical bugs will have the most impact on those who are new to such applications and are the most vulnerable, including older adults. As the aging population increases, there should be a focus to make software more usable in the long run. For years, Facebook used the motto "move fast and break things" (Baer, 2014, n.p.). This promoted rapid development with the goal of shipping software as quickly as possible and getting into the hands of users and sacrifices stability and a nuanced understanding of user needs. In an era that promotes "disruptive technology," a continued focus on the impact of this technology and the role it plays in the lives of its users is critical.

My first recommendation would be to understand the context within which software is being developed. This includes knowing who your users will be, but also involves understanding if it will be possible to interact with them directly in a representative way. If developing software for a specific user case (e.g. one specific client), then effort should be taken to involve users as often as necessary. A key element to this is multiperspectivity – a "basic prerequisite for cooperative work" (Floyd, 1989, p. 6). Maintaining user perspectives in the forefront of the development process involves looking at software "in the context of meaningful human activities" and "emphasizing the experience with software in use as primary level of concern."

Additionally, we can look at historical data to understand who may be using software after it has been released and understand their needs. In the case where software is being contracted for a specific client, this may be simple, but in the case where software

will be released for public use, it may be harder to predict the precise demographics. Musa (2009) discusses the use of operational profiles to create more user-friendly software. An operational profile "is simply a set of disjoint (only one can occur at a time) alternatives with the probability that each will occur. If A occurs 60 percent of the time and B 40 percent, for example, the operational profile is A, 0.6 and B, 0.4." Operational profiles were developed to focus the testing of large systems by prioritizing on the basis of use. I argue that these profiles can also be used to guide the process of software development as well by prioritizing the implementation of features and maximizing usability for the most heavily trafficked features first. This information could either be derived empirically through similar existing systems or estimated, likely with the help of users, upon the creation of entirely new systems.

How can software developers reconcile with the need to understand their users and the organizational context within which they reside, but also cope with the many challenges that are presented with user-centered design? The first step is recognizing that there does not exist a one-size fits all approach to software development. Setting hard rules with respect to user interaction is not helpful. In some settings, users will be quite homogenous, and in these cases adhering to stricter user-centered design may prove to be beneficial. One example of this may be an educational app for elementary schoolers. In other cases, with heterogeneous user groups and tight deadlines, it becomes increasingly difficult to derive meaningful data from user interactions. Economic constraints make interacting with user groups challenging (especially in the cases where applications are built by contractors), and even if doing so were feasible, applications with thousands of

users will not always have simple use cases. Therefore, even if such interaction were feasible, it would not necessarily provide representative data.

Conclusion

There are a variety of specific needs that need to be understood when developing software for cross-generational users. Some of these, such as physical usability concerns, are fairly well documented and understood. However, there exists little enforcement for these standards, likely due to economic pressure during the process of software development. While interacting directly with users may aid in understanding usability concerns and fill the current gap in existence knowledge regarding usability, it is necessary to understand the limitations that user engagement poses. It does not always provide a representative view of users' needs regarding an application, especially if the user population is diverse. Additionally, taking time to understand the specific needs of users can slow down the development of an application greatly. In these cases, it is important to weigh the possible benefit against the drawbacks of a delayed release (which could further delay preliminary usability reports).

The current strategy to "move fast and break things" and produce "disruptive technology" that reaches market as fast as possible that agile methodology can promote is harmful to users and unsustainable in the long run; patchwork solutions inhibit quality design and execution. Ultimately, there is no one-size-fits-all process for creating good software, but by understanding the limitations of agile methodology as well as the historical user needs to decide when further user engagement is necessary, developers can maximize their chances of creating usable software.

Works Cited

Anderson, M., Perrin, A., Jiang, J., & Kumar, M. (2019, April 22). 10% of Americans don't use the internet. Who are they? Retrieved from https://www.pewresearch.org/fact-tank/2019/04/22/some-americans-dont-use-the-internet-who-are-they/

Baer, Drake. (2014). "Mark Zuckerberg Explains Why Facebook Doesn't 'Move Fast And Break Things' Anymore". *Business Insider.*

Becker, S. A. (2004). A study of web usability for older adults seeking online health resources. *ACM Transactions on Computer-Human Interaction (TOCHI)*, *11*(4), 387–406. doi: 10.1145/1035575.1035578

Boehm, B., & Turner, R. (2003). Using risk to balance agile and plan- driven methods. *Computer*, *36*(6), 57–66. doi: 10.1109/mc.2003.1204376

Cockburn, A. (2009). *Agile software development: the cooperative game*. Upper Saddle River, NJ: Addison-Wesley.

Demographics of Internet and Home Broadband Usage in the United States. (n.d.). Retrieved from https://www.pewresearch.org/internet/fact-sheet/internet-broadband/

Deursen, A. J. V., & Helsper, E. J. (2015). A nuanced understanding of Internet use and non-use among the elderly. European Journal of Communication, 30(2), 171–187. doi: 10.1177/0267323115578059

Floyd, C., Reisin, F.-M., & Schmidt, G. (1989). STEPS to software development with users. *Lecture Notes in Computer Science ESEC 89*, 48–64. doi: 10.1007/3-540-51635-2_32

Geng, R., & Tian, J. (2015). Improving Web Navigation Usability by Comparing Actual and Anticipated Usage. *IEEE Transactions on Human-Machine Systems*, *45*(1), 84–94. doi: 10.1109/thms.2014.2363125

Goldman, S.L., Nagel, R.N., Preiss, K., 1995. Agile Competitors and Virtual Organizations: Strategies for Enriching the Customer. Van Nostrand Reinhold, New York.

Heinbokel, T., Sonnentag, S., Frese, M., Stolte, W., & Brodbeck, F. C. (1996). Don't underestimate the problems of user centredness in software development projects--there are many! *Behaviour & Information Technology*, *15*(4), 226–236. doi: 10.1080/014492996120157

J. D. Musa, "Operational profiles in software-reliability engineering," in *IEEE Software*, vol. 10, no. 2, pp. 14-32, March 1993. doi: 10.1109/52.199724

Kane, L., & Kane, L. (2019, September 8). Usability for Seniors: Challenges and Changes. Retrieved September 29, 2019, from https://www.nngroup.com/articles/usability-for-senior-citizens/.

Leiner, Barry et al., "A brief history of the internet," in *ACM SIGCOMM Computer Communication Review,* vol. 39, no. 5, pp. 22-31, October 2009. doi: 10.1145/1629607.1629613

Latour, B. (1992). Where are the missing masses? The sociology of a few mundane artifacts. In Bijker, W. E. and Law, J., eds. Shaping technology/Building society: Studies in sociotechnical change. Cambridge, MA: MIT Press, pp. 225-258.

Learning from Accessibility Research - AARP. (n.d.). Retrieved from https://assets.aarp.org/www.aarp.org_/articles/research/oww/university/RedishPresentation.pdf

Martin, M. W., & Schinzinger, R. (2010). *Introduction to engineering ethics*. Boston: McGraw-Hill Higher Education.

Morrell, R. W., Dailey, S. R., Feldman, C., Mayhorn, C. B., Echt, K. V., Holt, B. J., And Podany, K. I. 2004. Older adults and information technology: A compendium of scientific research and Web site accessibility guidelines. National Institute on Aging, Bethesda, MD.

UC Boulder (2019). Operational Profiles. https://www.cs.colostate.edu/~cs530/rh/section9.pdf

Warzel, C. (2020, February 4). The App That Broke the Iowa Caucus. Retrieved from https://www.nytimes.com/2020/02/04/opinion/iowa-caucus-app.html