User Interface Design & Software Accessibility

STS Research Paper
Presented to the Faculty of the
School of Engineering and Applied Science
University of Virginia

By

Makonnen Makonnen

May 08, 2020

Introduction

A user interface in the context of the field of software development and web applications is the medium in which a human interacts with the piece of technology they are using. How a user interface is designed is significant - as a badly designed user interface can prevent the intended audience from using it. With more and more devices using some form of computers , we are seeing a growing variety of interfaces to interact with them (Blair-Early & Zender, 2008) . Additionally, there is often a conflict of interest between the profitability of the company designing the application and the functionality of the application, for example when a designed chooses to create a flashy feature at the expense of usability to the end user. This isn't a trivial problem to solve and can sometimes be viewed as *tacit knowledge* due to its strong relationship to the arts. For example as a designer creating an application for an apartment building, what if I model it after *kotaku.com* , a site with a significant millennial user presence (Kilcullen , 2018) ? This could have a huge impact on the whole business. Residents, specifically those of an older demographic, could have issues navigating the site or simply understanding how to use it. This leads to issues with making payments or accomplishing tasks such as requesting maintenance assistance. In turn the business owner's reputability and profit can be negatively impacted.

To dive a bit into the *usability* aspect of user interfaces , the Institute of Electrical and Electronics Engineers (IEEE) created a high level metric. It’s called the Usability basics for software developer. It specifies the attributes as how easy the UI is to learn, efficiency (number of tasks one can perform per unit time) , how easy it is to retain skills learned, reducing error rate, and increasing user satisfaction (Ferre, Juristo, Windl, & Constantine , 2001). The guidelines described offer ways of maximizing all attributes and measuring how well one has accomplished them. Finally it also lays out various methods of analyzing users to gauge what type of UI a UI designer should create and how to conceptually design it. In order to address issues related to *user interfaces*, designers have realized the necessity of creating a design principle. Principles that can help create mindful UI that looks good, while also giving us some metric for measuring what ‘good UI’ or ‘bad UI’ is. There are a couple of philosophies as to how one should systematically approach this and although they might differ slightly in details and names, most accepted and successful one’s share the very same core ideas.

A well researched and applicable one is laid out in *Designing the User Interface: Strategies for Effective Human-Computer Interaction* in which Professor Ben Schneiderman lays out 8 usability heuristics for designing (Shneiderman, 1997). The first three rules describe metrics for how the UI should look towards the intended audience , exemplifying consistency, universal usability, and offering feedback. Rules four through eight describe what dialogue is with respect to user interfaces , prevention of errors, giving users control, and reducing how many things your user has to keep track of in their head.

While a framework such as this gives us a way to measure how well a user interface is designed and how to create our own, there are still some grey areas or rules defined that are very open ended. Exploring how user interface creators engage with their intended audience when designing and implementing their interface can reveal very interesting discoveries on the role of iterative design in user interfaces. Likewise, we can observe how this directly impacts people , specifically how accessible a site is.

Engagement with Users

The software development cycle is a timeline that is commonly used to implement large scale projects. It focuses on the customers of the project as well as the high level technical timeline. There are six steps to the cycle, the initial being the planning stage and the final being maintaining the product after it is released (Hussung, 2019). To reach the final step from the initial one developers must go through the *analysis stage* which defines the project goals (Hussung, 2019). This stage also includes addressing the technical details as well as conducting in-depth user surveying. How people think and how technology functions does not always match up (Norman, 2003). Seeing how users in this phase react to various things and how they perform in relation to what is expected allows designers to make educated assumptions on what the mental model of their target audience is. Using that collected information the team must *design* the product, keeping in mind usability and user interface standards.

Afterwards the team prepares for the final stage by implementing the designed product and testing it out. Testing at this stage consists of conducting technical testing alongside testing the product out with potential users to ensure UI and usability standards were met. If they failed, the team creating the product has to go back to the design stages to address the issue.

From the software development cycle described above we can see that the initial surveying of potential users, designing the product based on that, and testing if it was implemented well is key to a successful usable user interface. When designing the user interface and considering various cases of usability, it's critical to conduct audience analysis and engage with the intended users. In *Usability Basics for Software Developers*, IEEE gives a breakdown on how to conduct audience analysis. A developer should identify who the users are, what they want to accomplish, how they want the system to help them, and how the system will do that (Ferre, Juristo, Windl, & Constantine , 2001).

IEEE gives several methods to approach user analysis. One method is site visits. This is considered to be a go-to method as it is applicable to a majority of industries within software development (healthcare, music, etc.). What developers and designers do during this stage is they observe potential users conduct some task without using the product their team is about to create. This lets the team know common issues users have, what their mental model is, and how they generally behave. Using site visits gives a very organized method to collect information on users.

Focus groups and surveys are other methods in which designers get to interact with potential users directly. In focus groups a team organizes a discussion and they get to provide that group topics of discussion. These topics pertain to the product the team is attempting to create and refer to common problems or issues the participants of the focus group might have. Surveys essentially ask these questions, but in a more constricted manner as they are usually online or paper forms with set questions. A focus group has a bit of an advantage because it’s not restricted in the sense that one can ask follow up questions, but it is also more expensive since it requires finding a location to conduct it and getting the participants there.

Design with Intent

After collecting information from users , designers must apply it to their design. This starts with simply designing conceptually. This is regarded to be the key step in design (Ferre, Juristo, Windl, & Constantine , 2001). Typically designers can do this using paper/pencil or mock up tools available online. Here designers try to write out what the user will see and how they will interact with it. This step is supposed to be very high level and focuses on user-system interactions rather than what they see.

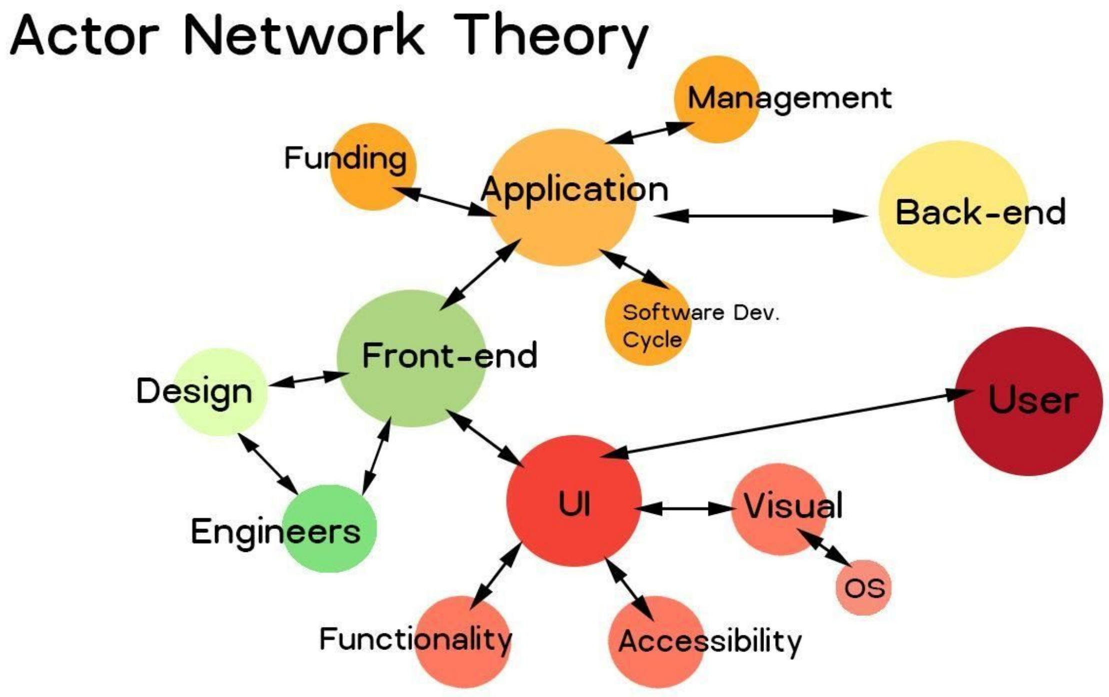
After conceptual design designers will visually design the product. Using the previous conceptual design step, designers define the interface very specifically. This includes, but is not limited to : color specifications, size of the fonts , how fast boxes will pop up. At first this might seem easy , simply choose colors that don’t *look ugly* , choose a font that you can easily see, etc. but there is a lot to consider with a designer's audience here.

For example let us explore two applications that have two completely different audiences - the College Board site vs. American Association of Retired Persons. The College Board is commonly used by high schooler's when applying to colleges ("The Collegeboard Homepage", n.d.). It has a database of an individual's college entrance exam scores as well as a financial aid evaluation tool called the CSS. When entering the website a user is greeted with very stylish elements. There is heavy usage of semi-playful small fonts , very light colors, and a prevalence of social media links. In comparison the AARP site has more reserved colors, darker and larger font usage, and a more rigid structure to it with almost everything being accessible from the landing page ("AARP Official Site", n.d.). It's evident that the designers of both sites carefully looked at their audiences and applied usability and user interface standards to meet their needs. Following well defined standards to produce a good looking product is wonderful and all, but what really matters in the end is the accessibility. If the interface is what is preventing customers from interacting with one's product , then there was a big problem in either the initial user surveying or during the design stages.

Accessibility

According to Mads Soeggard "A design is only useful if it's accessible to the user: *any* user, anywhere, anytime"(Soeggard, n.d.). Soeggard describes aspects of accessibility and I'd like to focus on visual and auditory accessibility (Soeggard, n.d.). He defines visual accessibility addresses those who have vision problems like blindness, vision issues and it can be extended to those who can be visually affected by lights - like those with photosensitive epilepsy. Soeggard then describes auditory accessibility by saying it addresses those who have hearing problems due to some level of deafness. If a website is not accessible by some of these standards there are legal repercussions people can take, for example, there is the EU Web Accessibility Directive. The directive states “In the context of this Directive, accessibility should be understood as principles and techniques to be observed when designing, constructing, maintaining, and updating websites and mobile applications in order to make them more accessible to users, in particular persons with disabilities”. This can cost a company a large amount of money. In the US the ADA (American with Disabilities Act) allows for similar actions which was shown earlier this year when the Supreme court allowed a blind man to sue Domino’s over the site accessibility (Tuckerhiggins, 2019).

Applying Actor Network Theory



This is the Actor Network Theory I am proposing to approach the topic of accessibility within software. By observing certain interactions between the many actors listed - I think we can see why accessibility is a challenge in creating applications. In terms of how I broke up the diagram I primarily focused on the user interacting with an application and how that flows. You initially have some concept of the application which requires management and funding. Then it gets kicked into production via the software development cycle I describe earlier in this thesis. From there the front-end engineers create the user interface that the user interacts with. I added the Back-end non-human actor because although the user doesn't interact with it , most modern web applications need some form of a back-end to function (the user doesn't interact with it directly though). I also added visual + OS actors because depending on what device you use the visuals could be different. For example, using collab on your phone and computer has two different visuals and this is dependent on the operating system / device you're using.

ANT Diagram - Application

The application itself is some combination of a user interface or front-end that the user interacts with and a back-end , which is where data will be stored and where the logic of what is displayed might exist. Before any major application can become accessible to users it must acquire funding for multiple reasons: maintaining the application, paying for the servers that will host the application, and paying for the engineers working on the application. Funding can be acquired by various means of personal savings, asking people directly (think sites like goFundMe) , loans, but the major players usually go the route of using Venture Capital firms.

At venture capitals an idea is pitched and tweaked based on the firm's comments. I think this has an impact on accessibility because although we know focusing on accessibility is important it might not be the most profitable thing to work on. I think introducing the conversation of accessibility at this stage will have the biggest impact. Since V.C.s have input on features of the application before funding an application idea , asking about how accessibility will be tackled by a team and asking for more if it’s not enough can really push for accessible applications from the beginning instead of late in the implementation. This also allows for the software development cycle to emphasize accessibility rather than ignoring it.

ANT Diagram - Front-end & UI

The front-end is **everything** the user sees and interacts with. The building blocks of web applications are HTML, CSS, and JavaScript. HTML is like the bones of any application. It’s what our web browser can read/see and understand. CSS is like the skin. It’s styling, colors, etc. JavaScript is like the limbs or muscles. It can interact with HTML and CSS to execute some logic like change this color from red to yellow or calculate what 1+1 is. The engineers who work on the front-end are called front-end engineers and they usually work with a UI/UX design team. That team designs the look on paper and the engineers implement it via HTML, CSS, and JavaScript.

The critical step here is the design component. Whereas bad engineering implementation can have side effects (i.e. application not being as smooth as expected, etc.) if the engineer implements a design perfectly and users have accessibility issues with it, the issue is due to the original design - not how it was implemented. Designers understanding who their target audience is, who might potentially have issues with X vs Y designs, and experimenting is critical. On top of this understanding what designs are realistic to implement so no corners are cut is important. Although we see amazing designs and UIs coming from some popular companies - a lot of the UI we use in our daily lives is very outdated and it’s fairly easy to see that the design stage wasn’t in-depth or the focus for the managers of the application. From sites like SIS to a lot of rent payment websites in Charlottesville, we see common issues like very tiny fonts that’s not readable , ignoring mental models of the users, etc. Tackling the issue at the design stage would have resolved a lot of these accessibility issues when the front-end engineers implemented the user interface.

ANT - Summary

Currently the general pipeline consists of some idea -> raising money for that idea -> then throwing it into the software development cycle under some management. In the software development cycle (in context of user interaction) we have a front end engineering team consisting of designers and engineers. The designers draw out the interface and the engineers implement it in code. The user then interacts with the interface looking for some desired functionality on their respective device. Depending on how the interface was designed, it's either accessible or not for the user.

As discussed in previous sections - introducing new actors in the diagram or items in the pipeline can significantly have an impact on accessibility. Introducing an accessibility focus in the initial funding stage by V.C. can really promote behaviours and economic incentives for engineers. Then at the design stage , adding various ways to ensure accessibility before implementing it. This could be done by testing various users ranging from target demographics to people with disabilities to make sure that things are accessible in action.

Conclusion

A lot of the decisions made during UI design have to do with Iterative Design Decisions. How to make something simple yet functional? On top of this - how does one make it accessible? What are the costs related with sacrificing things like accessibility and what are the costs with trying to hit every point making a product accessible. It's fairly challenging designing the '*perfect*' user interface, but using various metrics like the IEEE guidelines and Schneiderman's rules designers can push towards a product that has a usable and accessible user interface, that is free from discrimination, and gets it's intended job done. On top of this by looking at software development under an Actor Network Theory schematic we can obviously see some weaknesses or flaws that contribute to this pertinent issue. Introducing new actors that change behaviours early in the development stage + pushes for more user testing in context of accessibility can increase accessibility significantly. We live in a society that is experiencing rapid technological advances and with new ways to use them . By focusing on user interface design intensely in computing related education as well as software bootcamps, I believe major problems can be mitigated.

References

AARP Official Site. (n.d.). Retrieved from https://www.aarp.org/.

Blair-Early, A., & Zender, M. (2008). User Interface Design Principles for Interaction Design. *Design Issues*, *24*(3), 85–107. Retrieved from https://ieeexplore.ieee.org/document/6792480?denied=

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, October 26). Retrieved from https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32016L2102.

Ferre, X., Juristo, N., Windl, H., & Constantine, L. (2001). Usability basics for software developers. *IEEE Software*, *18*(1), 22–29.

Hussung, T. (2019, August 26). What is the Software Development Cycle? Retrieved from https://online.husson.edu/software-development-cycle/.

Kilcullen, S. (2018, September 25). Ranking Websites by Demographics. Retrieved from https://www.quantcast.com/blog/ranking-websites-by-demographics/.

Shneiderman, B. (1997). *Designing the user interface: strategies for effective human-computer interaction*. Boston: Pearson.

Norman, D. A. (1993). *Things that make us smart: defending human attributes in the age of the machine*. New York: Basic Books.

Soegaard, M. (n.d.). Accessibility: Usability for all. Retrieved from https://www.interaction-design.org/literature/article/accessibility-usability-for-all.

The Collegeboard Homepage. (n.d.). Retrieved from https://www.collegeboard.org/.

Tuckerhiggins. (2019, October 8). Supreme Court hands victory to blind man who sued Domino's over site accessibility. Retrieved from https://www.cnbc.com/2019/10/07/dominos-supreme-court.html.