CS 4712 User Interface Engineering

MODULE_2 – TOPICS

Conceptualization of User Interface Design

Contents

Module_2 - Conceptualization of User Interface Design

MODULE_2 – TOPICS

CONCEPTUALIZATION OF USER INTERFACE DESIGN

2.1 - Introduction and Background

As technology and its uses have advanced over the years, so has the ways we interact with technology. Touch screens were uncommon in the early 2000's. Now, you can see and interact with them almost anywhere, and phones are now created with touch screens in mind. The overall thought process for creating user interface design includes how make the application or device work faster as well as more efficiently. In this topic, we will be looking at the conceptualization of user interface engineering and explore the thought process behind it as well as its evolution. Before going forward, two topics must be defined: Conceptualization and User Interface Design.

Conceptualization is the process of clarifying and developing of terms, arriving at precise definitions ("Conceptualization excerpt", 2012). Conceptualization is the clear and concise way of applying definitions to the various design concepts, so that nothing becomes confused when carrying out said concepts. While most people will have different perspectives at the time a concept is presented to them, the conceptualization process should at least have ideological differences cleared up.

User interface design is the way to design interfaces, usually in software applications, to make the product easy and pleasurable to use. UI design usually refers to graphical interfaces but may also include voice-controlled interfaces as well. Any well implemented and sophisticated product should satisfy the vast number of characteristics that are involved in Graphical User Interfaced (GUI) design without interrupting the overall usage of the product at hand. It is also important to provide an in-depth exploration of the coding languages used within user interface design. Since UI design is so important in the context of Computer Science and Software Engineering, a discussion of GUIs and their inner workings will be important.

One must also understand the principles incorporated into UI design and how well those principles will work within a certain context. Explored are best-case and worst-case scenarios, how these principles interrelate with one another, and finally what are considered top priorities within UI design.

So how does conceptualization improve UI design, and what is the overall step-by-step process used for conceptualization? What models are used in this process? This module will take into account what needs to be consistently completed in order to release a fully formed UI design within the context of a program.

2.2 - Problem and Design Environments and Models

There are a few problems associated with GUI design, namely how to make elements appear appealing, how to make elements' functions obvious, how to properly plan architecture, and how to make correct requirement assumptions (Jones, 2018).

When looking at how to make elements appear appealing, GUI designers often face the problem of whether or not to use new elements that continue to become available. While this might help with staying on top of trends, the majority of users may not know exactly how to interact with these newer GUI elements.

This leads to confusion and frustration (Jones, 2018). The key to resolving this lies in making sure that elements perform the function that they appear to perform. If a GUI designer places a button on a page, it should in fact perform some action. It should not be used simply to highlight an important word or title on a page. Along those same lines, text should not look like a link if it is really just important text being highlighted. In both instances, users may click on the button or link and expect an action to be performed, leading to confusion if nothing happens. This same ideal goes into making sure there is a visual confirmation or error displayed when a user clicks a button. If, perhaps, a user clicks "submit" on the form and does not receive a message as to the status of the request, they may continue to press the submit button in hopes of receiving that message (Jones, 2018).

The issues concerning buttons and links usually occur due to poor design architecture. This can lead to a myriad of problems. If a design is not well architected, developers may simply begin throwing elements onto a page without thinking about how they make affect the page as a whole. This can lead to significant rework and missing deadlines. A better approach is to first think about the goal of the GUI as well as the overall look and feel that is trying to be achieved. Often, making things such as wireframes beforehand can assist in ensuring a cohesive design (Jones, 2018).

Another challenge when it comes to GUI design is that requirements are not always clear. Usually, a client has an idea of what they want the end product to look like or the capabilities they want it to have. However, these thoughts are mostly expressed in terms of business processes or are based off of their current system. That does not mean it is always the best solution to their problem. It is imperative that designers get accurate and thorough requirements from the stakeholders before beginning the design and architecture of a GUI. It often helps to create a wireframe and have the client review it before moving onto the actual development part of the GUI (Jones, 2018).

Wireframes are just one of the design options available when attempting to create models. Designers can also choose to create prototypes to demonstrate their ideas to developers and end users (Cahill, 2019). Carl Cahill gives a pretty extensive list of tools available for both of these design options. Some of the tools for wireframing are Mockflow, Balsamiq, and Axure. Some tools for prototyping are Sketch, InVision Studio, and Proto.io. A more extensive list as well as descriptions for each product can be found at the website detailed in the sources below (Cahill, 2019).

These two design options, wireframing and prototyping, are often mistaken by designers as being one and the same. However, they both serve different purposes and are usually utilized at different phases of the design process (Fanguy, 2018). Wireframes are used earlier in the design process and are more high-level. They show the developers as well as end users what content will appear on the screen, how text and images are formatted, and other visual design specifications. Wireframes do not, however, allow a user to interact with or test the graphical user interface (Fanguy, 2018).

Prototypes come into play a bit later than wireframes. Prototypes also show the layout of content and other design specifications, but they can also be interacted with by a user. This is not to say that all of the end result's functionality will be available, but a user can test out navigating through the basic GUI when presented with a prototype (Fanguy, 2018).

When it comes to designing graphical user interfaces, it often helps to first develop models off of which to base the actual programmatic development. This approach allows designers to show developers what a design should look like or contain without getting bogged down in the actual programming of the GUI

(Meixner, 2013). Designing a GUI based on models also helps shave a significant amount off of the overall development time. For many applications, the design and implementation of the GUI accounts for about half of the total development time, as well as taking a little over a third of total maintenance time (Meixner, 2013).

By creating models such as wireframes, designers can focus their efforts on designing the entire system without having to wait for each portion to be completed before moving on to the next. This also leads to a better end product, as developers can see the desired result of the cohesive application GUI before they even start programming. This enables the developers to program portions of a GUI while having the desired look in the backs of their minds (Meixner, 2013).

There are many other benefits of model-based GUI design. These benefits include an implementation that is closer to the design requirements; allowing multiple stakeholders to give feedback on the overall design of the GUI, producing quality architecture; allowing for a higher level of abstraction, improving the quality of development; and making maintenance an easier task (Meixner, 2013).

2.3 - Characteristics of Graphical User Interface

The Graphical User Interface, or GUI, is one means through which end users interact with software. Any particular action they take in the GUI usually has a corresponding action that occurs behind the scenes. According to Brainkart, a GUI consists of a sophisticated visual presentation, pick-and-click interaction, a restricted set of interface options, visualization, object orientation, use of recognition memory, and concurrent performance of functions (Characteristics of the Graphical User Interface).

What makes a visual presentation sophisticated? If we first look at an unsophisticated visual presentation, we will be able to better understand the answer to this question. An example of a simple visual presentation in a GUI would be the native calculator app on a computer. This application (typically) displays 9 digits, a decimal, mathematical operators, a clear button, and a display line where the results of the operation are shown. This is simple in terms of a GUI, because each button represents one function. A user is not able to select from a list of operations when clicking a single button, they are not able to scroll within the application, and they cannot make a sub-window open.

On the other hand, consider an Integrated Development Environment, or IDE. IntelliJ IDEA, for instance, has a very complex visual presentation. At the start, there are options for creating, opening, or importing a project. Once a user chooses one of these options, a new window appears with either the contents of a saved project or a blank project. One can then view the file structure of the project, open individual files, edit individual files, enter commands in the terminal, run a program, and more. The list of possibilities goes on for quite some time. Unlike the native calculator application, a user can do things such as scroll vertically and horizontally, open new sub-windows, or choose from a list of actions by clicking a single parent button. For a GUI to be considered a sophisticated visual presentation, it must provide many different possible ways for the end user to interact with it. These can include, but are not limited to, windows, menus, and icons (Characteristics of the Graphical User Interface).

Looking next at pick-and-click interaction, let us first define this in a more granular way. According to Brainkart, “To identify a proposed action is commonly referred to as pick, the signal to perform an action as click” (Characteristics of the Graphical User Interface). Going back to the two application examples above, each of these GUIs provides the user with pick-and-click capabilities. While one may be more sophisticated than the other, both successfully fulfill this aspect with respect to the end goal of their applications.

Next, a designer may consider a restricted set of interface functions. The main idea behind this concept is What You See Is What You Get (Characteristics of the Graphical User Interface). This means that a user has a finite set of options available to them, and that each available option is clearly defined. Looking once again at the calculator example, while simplistic, each option is very clear about its corresponding function. This clarity is provided through the use of symbols or icons that describe what happens when a user clicks one of the icons.

Conversely, the IntelliJ IDEA can potentially be confusing. It provides a large list of functions, some of which can be unclear when it comes to what actions are performed as well as when they are applicable. A complex GUI such as this one attempts to make what can and cannot be done clearer through the disabling of certain options at specific phases of the GUI lifecycle. For instance, a user cannot always choose to run a program; there must first be a program to execute. However, a user can always choose the help option, which will open another window with frequently asked questions.

Both of these interfaces provide different methods and levels of restricting interface functions. The methods they employ are largely driven by the complexity of the individual application and its corresponding GUI.

Visualization is an aspect of the GUI that can make or break a program in the eyes of the consumer. If a GUI is cluttered with a large volume of text or functions that are difficult to understand, a user may be less inclined to use the software. Applications can avoid this mistake by keeping their GUIs sleek and concise. This does not mean removing sophistication, but rather finding better ways to display certain pieces of material, such as a sidebar that can be expanded or collapsed to show additional functions based on where the user is at in the application's lifecycle, or a graph, figure, or image, to make what would have been complex data more readable for the user.

Typically, when we think of object orientation, we think of programming languages. However, this concept can also be applied to GUIs. When looking at a GUI, the individual sections that users can see can be thought of as objects. Each of these objects can have sub-objects which define the details of the parent object (Characteristics of the Graphical User Interface). If we look again at the calculator example, we can think of the area that contains the buttons as one object and the area that displays the output as another object. These objects are both children of the parent calculator object and have children that define what they do and show. The object that contains the buttons is made up of sub-objects for each of the buttons. These sub-objects are responsible for the icon or symbol that represents the button as well as the function that is executed when one of the buttons is clicked. The object that contains the output is made up of a sub-object for the display section and a sub-object for the output data. These object control formatting that is visible to the user.

Moving on to the use of recognition memory sparks the debate of whether functions should be tucked away in tab or menus, or if they should all be displayed in the same place. There are pros and cons to each of these ideas. If we decide that hiding all of the complex actions in a submenu is the best idea, we will be left with a much more concise main screen. However, this will require the user to have a knowledge about where they need to go in the GUI to perform specific actions. If the categorization of these functions is not intuitive, we risk the user having difficulty learning the application and subsequently looking for an alternative. On the other hand, if we decide that we should have all of the functions visible no matter the point in the application lifecycle, we risk exposing a cluttered GUI that the user cannot stand to navigate. This once again leads to discontent and the risk of the user moving on to an alternative solution.

The best way around either of these worst-case scenarios occurring is to present the user a GUI that melds the two ideas together. For instance, if there are functions that should always be available, it would be best to have them persist on the GUI throughout the duration of the application. If there are functions that are used less frequently, they can be hidden in sub-sections or menus to reduce the clutter of the GUI.

Finally, designers must contend with the concurrent performance of functions. At any time, a user can choose to perform an action which either causes another action to execute or causes some data to need to persist beyond the current transaction (an example would be copying information to the clipboard and storing it until it is overwritten) (Characteristics of the Graphical User Interface). To allow the user to perform these actions, they must be allowed to execute in the background while the user is still interacting with the GUI in a different manner. Otherwise, the user is sitting and waiting for the first action to complete, effectively wasting time that could be better spent elsewhere.

In conclusion, a GUI consists of many characteristics, some of which are quite complex. Designers must consider all of these characteristics to ensure that users have a pleasant experience.

2.4 - Guidelines, Theories, Principles, Models and Frameworks

When creating a GUI, it's important to keep in mind the user that will be using the application. In general terms, the GUI should scale with the level of computer competency of the average user. We can see this in action in the most familiar applications. For example, Facebook has a very minimalist design approach, with buttons that stand out so that the average user can properly navigate the website. Meanwhile, Stack Overflow (the programmer help website) is condensed with information that clutters the page, with many options hidden inside other options. This is, in part, due to the average user of the website having more experience with technology than a user of Facebook.

While complexity is a major component of user interface design, one should also take into account how visually appealing the interface is for the end user. A website or computer application that is visually unappealing to the end user is one that they may choose not return to.

When creating a user interface experience, one should always create a clear and concise outline of the purpose for the user. This can be illustrated by outlining the steps a user will take to reach their goal or "destination": form the goal, form the intention, specify the action, execute the action, perceive the system state, interpret the system state, and evaluate the outcome.

When forming the goal, the user predetermines ("almost always before interacting with the interface") what their goal is when using a product. This can often be as simple as accessing a help page or as complex as creating a full presentation through the use of that program. "Form the intention" focuses on what intention users have when they begin to use a product. Next, the user specifies a logical action to take. Execution of the action follows through on what happens when the end user specifies that action. A common example would be a user clicking on a button leading to another webpage. "Interpret the system state" refers to how the user will respond to when the system changed because their actions. Finally, evaluate the outcome refers to how the end user will evaluate whether their action lead them closer to their goal or not. This is the foundation for all well-designed user interface design.

A framework in user interface refers to the software tools used to build software programs that run on the web. It's normally used by web developers, so a front-end and back-end framework can be built using HTML, CSS, and JavaScript. There exist other frameworks, however, for multiple languages. Frameworks

are often used so that a developer does not have to actively reinvent the wheel for every new project. In addition, frameworks provide longevity to applications by making them easier to maintain and extend. Good framework design also enforces good coding practices and provides the developer with ample community support. This usually leads to less time trying to debug code, as the community based around the framework has already found common solutions.

It's important when deciding to use a framework for a user interface application that one chooses the framework that best fits the need. Scope is an important concept to keep in mind for developing web applications. For small applications, a framework might over complicate and obfuscate the code, often leading to longer time decoding and bug fixing than creating a smaller web application. A larger web application that will utilize many different functions in the application, that will use different utilities, or that will have repetitive code often will benefit greatly from the usage of a framework. It's important to note that some companies choose the path to create their own company-maintained framework. This has the benefit of being more refined and easier to understand than other frameworks at the cost of serious time commitment. This remains a viable option for anyone looking to use a framework that is also not in a more common or practical field of operation for web applications (such as data science or artificial intelligence).

2.5 – General Design Principles

While there are multiple ways to design UIs, there should be some core principles. There are arguments over whether, how many, and in what order principles are needed. Some would argue for 4, some for 6, while others may go as high as 19. One scholar even argues through several different models through which one could start a web UI (Constantine 2001). Through these various sources these principles will be explained as precisely as possible. These examples will come from multiple people who are skilled with design and do these types of projects as a professional career.

Before listing out and defining all the principles of User Interface Design, the most important principle needs to be defined and listed: clarity. Having clarity is something that most sources can agree on. How clarity fits into the overall picture may differ from one author to the next. Clarity can be the first step in a very large path in order to complete the task at hand (Porter N.D.). Having a clear design is important because someone who is frustrated with the design is going to not want to have anything to do with the technology involved. Other objectives are to encourage and enable interaction with the user. Being clear should help minimize any confusion that the user has when interacting with these UIs. Through thorough reading, clarity should not be something that designers take for granted. Using a UI should not be punishing but rather something that feels seamless and invisible. Since most of our world is connected by technology, it would be important to consider whether the UI can keep the attention of whoever is using an application or feature. For example, the average user only stays on a webpage for a very short period of time, but if a page can keep a user engaged for 30 seconds, the chances increase that the user will stay for an extended period of time (Nielson 2011).

For most electronic hardware with screen interfaces, only containing information on one screen is enough. A logical next step as well certain amounts of on-screen clutter should inspire us to see the first principle of clarity. It is important to have a logical next step as well as keeping certain actions secondary. Without a logical step next within the design, some users may be baffled and or confused on what to do and how to complete it. Some controls should remain secondary just in case a more advanced user wants to change it up (Babich 2018). Also, secondary actions should be only recommended to those who are seasoned users of the design. At times when designing a system, we should ask whether a step is really necessary in order to

complete the overall task at hand. Remember that appearance should also be apparent with its behavior (Porter N. D.). Keeping a consistent behavior is valuable to any UI design and makes the users more comfortable.

Overall the great design is indeed seamless and should not be noticed by the user. As the old idiom goes, "The squeaky wheel gets the work." Well-designed UIs should neither anger nor frustrate the consumer. While UI design may never be perfect, great UIs can be created using the principles and modules in various conditions and formats. These principles should help guide the user to a great experience with whatever the final product may be.

2.6 Concise Summary

2.6.1 In the introduction we introduced the topic and the general definitions describing how conceptualization interacts with User Interface (UI) design. We also explained what can be used for UI design and what could influence it.

2.6.2 When designing a Graphical User Interface, designers face many challenges and problems. Specifically, the designers must account for many things such as architecture and overall appeal. To best accomplish this, many designers prefer to work in a particular design environment and create wireframes and prototypes. The other way designers attempt to mitigate the problems associated with Graphical User Interface design is to follow a model-based format.

2.6.3 Characteristics of a Graphical User Interface include quite a large range of items. The list includes how elements are visually presented to users, how elements have corresponding back-end actions with which they are associated, how processes can be performed concurrently in the background, and many more. These characteristics highlight a couple of key points to consider when designing a Graphical User Interface. Furthermore, comparing two different GUIs can reveal the impacts to each of their overall designs. More complex concepts apply more heavily to more complex interfaces, while they do not apply quite as heavily to simpler interfaces.

2.6.4 When designing Graphical User Interfaces, it is important to remember that following some basic guidelines allows designers to create applications that meet the end user's needs, as well as keeping in line with best practices from around the industry. There are many different views and theories on what the best practice in specific cases is, so the final decision still comes down to the designer on which of these to follow. Frameworks enable a designer or developer to pick a large set of best practices to follow, as opposed to picking and choosing specific practices one by one.

2.6.5 User Interface design is based on many different underlying principles. Chief among these principles is one rule in particular: clarity. Ensure that the interface has a clear purpose. Designers should also ensure that actions available on a page also have clear purposes. When taking the clarity principle to heart, designers will naturally find themselves following other principles of good user interface design. In the end, designers must make sure that the interface is designed in a way that will not frustrate or anger users.

2.6.6 Important literature:

Seven Unbreakable laws of User Interface Design – this article covers the basic laws that every user interface should follow. These laws are briefly discussed throughout the course of this paper, while elaborating on the examples given throughout the paper (citing Google as the most common example of good user interface design).

User Interface Design Guidelines: 10 Rules of Thumb – This article provides some additional guidelines one should follow when creating a user interface. What separates this article from others is its focus on designing an interface that mimics real-world language that would be used by the user and lowers the cognitive load on a user. These are core aspects often lacking when many corporations create their own user design. In addition, this article emphasizes the minimalist design discussed throughout the paper. As minimalism is on the rise, designing a user interface design around minimalism is often encouraged. Lastly,

documentation is briefly explored, a core aspect of not only user design, but any coding for major projects. It's important to heavily document any major project that will take place.

UI Design Do's and Don'ts by Apple – A web article from Apple is not to be ignored. Apple, widely praised for their intuitive design, their sleek UI, beautiful look, and their simplistic design, is widely known as the king of user interfaces. This article covers UI design for iOS devices (mobile devices). This is extremely useful for developers, as most UI guidelines are broader and less specific, a hindrance to most mobile developers. iOS and other mobile platforms have to follow different guidelines from regular personal computers due to the limitations of the platforms. Because of this, we see how developers can use these limitations to their advantage, as noted in the article.

Principle of Consistency and Standards in User Interface Design – This article comes from the same author as the 10 Rules of Thumb article. Unlike his previous article, this one emphasizes the importance of consistency and standards by providing detailed reasons why we follow those guidelines. In addition to this, the author provides various examples from different websites showing why some corporations succeed and fail at keeping consistency between their different products. In addition to this, he provides five different ways that any developer or corporation can easily achieve consistency in their user interface design. Lastly, color and the usage of bars in the user interface are heavily emphasized.

Principles of User Interface Design – This article is focused on the fundamental principles that are a large part of any form of UI. Topics such as using the interface to conserve attention and creating an inline so that the interface is easily learnable for the user are discussed. Designers must keep in mind the fundamental principles of UI, as these principles held guide any design in the right direction.

2.7 - Extended Resources

Descriptions & Links

1. Apple Inc. (n.d.). UI Design Do's and Don'ts. Retrieved from https://developer.apple.com/design/tips/

2. Babich, Nick The 4 Golden Rules of UI Design retrieved July 22, 2019 from https://theblog.adobe.com/4-golden-rules-ui-design/

3. Porter, J. (n.d.). Principles of User Interface Design. Retrieved June 10, 2019 from http://bokardo.com/principles-of-user-interface-design

4. Nielson, Jakob How Long Do Users Stay on Web Pages (2011) Retrieved June 10, 2019, from https://www.nngroup.com/articles/how-long-do-users-stay-on-web-pages/

5. What is User Interface (UI) Design Retrieved June 10, 2019, from https://www.interaction-design.org/literature/topics/ui-design

6. GUI Design Guidelines. (2019). Retrieved June 10, 2019 from https://www.comp.nus.edu.sg/~cs3249/lecture/GUI%20design.pdf

7. The Theory behind User Interface Design. (2002). Retrieved June 10, 2019 from https://www.developer.com/design/article.php/1545991/The-Theory-Behind-User-Interface-Design-Part-One.htm

8. Conceptualization excerpt (2012) Retrieved June 10, 2019, from https://2012books.lardbucket.org/books/sociological-inquiry-principles-qualitative-and-quantitative-methods/s09-02-conceptualization.html

2.8 - References

Cahill, C. (2019, January 21). 20 best UI design tools. Retrieved July 8, 2019, from https://www.creativebloq.com/how-to/20-best-ui-design-tools

Constantine, Larry L. & Lockwood, Lucy A.D. (2001) Usage Centered Engineering for Web Applications Retrieved July 22, 2019 from https://pdfs.semanticscholar.org/6cce/edd33cbe74008364875c05eea3b4e1be91af.pdf

Fanguy, W. (2018, March 21). Wireframing vs. prototyping: What's the difference? Retrieved July 8, 2019, from https://www.invisionapp.com/inside-design/wireframe-prototype-difference/

Jones, T. (2018, June 17). 5 Common Challenges Faced By UI Designers. Retrieved June 22, 2019, from https://usabilitygeek.com/5-common-challenges-faced-by-ui-designers/

Meixner, G., Calvary, G., & Coutaz, J. (Eds.). (2013). Introduction to Model-Based User Interfaces. Retrieved June 22, 2019, from https://www.w3.org/TR/mbui-intro/

Vukovic, P. (2018, January 24). 7 unbreakable laws of user interface design. Retrieved from https://99designs.com/blog/tips/7-unbreakable-laws-of-user-interface-design /

Wong, E. (n.d.). User Interface Design Guidelines: 10 Rules of Thumb. Retrieved from https://www.interaction-design.org/literature/article/user-interface-design-guidelines-10-rules-

Wong, E. (n.d.). Principle of Consistency and Standards in User Interface Design. Retrieved from https://www.interaction-design.org/literature/article/principle-of-consistency-and-standards-in-user-interface-design

Characteristics of the Graphical User Interface. (2019). Retrieved June 8, 2019, from https://www.brainkart.com/article/Characteristics-of-the-Graphical-User-Interface_9007/

Acknowledgement: The initial preparing of this document was supported by CS 4712.