Megan Cunningham

INFO 608 HCI

# Week 1 DOET Digest

In this digest, we will discuss Don Norman's, *The Design of Everyday Things*. In the first two chapters, he emphasizes the importance of human-centered design (HCD)[1] and how designers can employ psychological concepts into their designs. One of the psychological concepts that intrigued my interest was around feedback and how feedback in software can either support the people's feelings of self-blame or support their use of devices.

I recently received a gift called the Nutr; which allowed me to make my own non-dairy milk. As I was reading the first two chapters of Don Norman's book, I found myself thinking about this product. While this product has a sleek visual design, as you can see in Figure 1, where I experienced a positive visceral reaction, it lacked some key signifiers and feedback. The instructions provided on how to use the machine failed to mention how the button panel worked on the machine. On first use, I clicked 'Start' to turn the machine on and then tried to select 'Warm' to make oat milk. The temperature buttons were all on the same, smooth, panel as the 'Start/Cancel' button and thus I had created a concept model that assumed all the lit-up words were functional buttons. However, that was not the case. At some point I kept hitting every word on the panel until eventually I noticed that if you hit 'Start' again that it would go to the next lit-up word on the panel and that word would blink indicating that it was selected. From then on, the machine was easy to use, and I was on my way to perfecting my oat milk recipe.

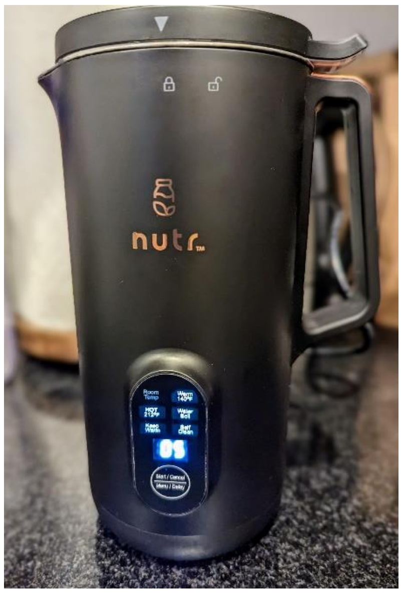

*Figure 1: The Nutr Machine turned on with the option panel.*

The reason that I kept thinking about the Nutr, was that it was easy to relate my experience with the important psychological concepts Norman describes. For example, the machine beeps and shuts off if the lid isn't secured properly and there's a line that signifies what the max fill line is for liquid. It made me wonder, with all of the other design considerations – Was I supposed to assume the correct mechanism or were the instructions inadequate?

In Norman's book, he describes how people often blame themselves when the design is at fault (Norman, 2013, chapter 2). If that is the case, what causes people to blame themselves? Are there cases where people believe that design is at fault? For instance, with the Nutr machine, I initially thought I was to blame as I am not the best

[1] Human-centered design: Design approach that focuses on accommodating human behavior, needs, and capabilities. (Norman, Chapter 1, 2013, Human-Centered Design)

with following instructions exactly. However, when I read the instructions again, I found that they did not signify that I needed to press the start button continuously. Thus, while I initially blamed myself, I later realized that the design was at fault for not providing thorough instructions. I could see how some people may think that because the instructions were missing, that the designers may have thought that everyone would know how to use the machine and that they were at fault for not being 'in the know.' As people have varying degrees of life experiences, I theorize that those experiences can determine how you react to design mishaps.

For example, I have developed software applications and know that issues that occur are rarely true user error but rather application faults. Having to troubleshoot and fix issues has influenced my mindset to understand the limitations of machines and their designs. Thus, I feel that I am predisposed to blame the machine. On the other hand, those without that background may not understand what they are doing wrong or may avoid complex applications or devices that they feel they aren't capable of learning. While they are most likely capable of learning a system, it would likely take more time than those that designed it or have used a similar tool in the past. Additionally, system errors and feature bloat may also discourage people from trying to use it.

Ideally technology is created to make people's lives better and simpler; yet it often can become increasingly complex (Norman, 2013, Chapter 1, The Paradox of Technology). In the world of software development, we have something called scope creep. This usually occurs when the client of the software being built requests features that aren't core to the system and beyond the initial scope. This increases the workload for the engineers and may also end up adding unnecessary complexity to the product. When people end up using the system, they are more likely to get easily confused. This can lead to poor user experience. Additionally, quality may suffer if the budget or timeline are not adjusted in accordance with the scope changes. This could to unexpected outcomes when people interact with the system. Fortunately, Norman offers advice regarding design to help avoid design pitfalls and user self-blame.

One of the six pieces of design advice that Norman provides is to "assume that what people have done is partially correct, so if it is inappropriate, provide the guidance that allows them to correct the problem and be on their way" (Positive Psychology, para. 5). This is a very important piece of advice for a few reasons. One, when a design provides feedback in way that enables a person to figure out how to adapt their concept model, people are less likely to give-up on the system. Two, the system avoids user self-blame as it shows the user that others may also encounter these same impedances. My discussion question for the week focused on how error messages can be critical in Software Security. I wondered why Norman recommended removing all error messages. Jamie Robinson commented, mentioning it was possibly because error messages seem to imply an issue with the user rather than the system and thus reinforce self-blame (Robinson, 2023). I think this is a perfect example of how software developers are often not informed of design advice and often don't think about things from a user-experience lens.

When I wrote software programs, I thought that error messages were useful to the people using the system. However, I did not realize that they are more useful to the developers. Most modern-day software systems do not need to show the user the actual error message and can display something else to the user. This could be in the form of feedback. As failures are likely inevitable, rather than displaying a useless error message to the user, a feedback mechanism could be provided to help them fix the issue. To ensure proper feedback is incorporated into designs, there needs to be a stronger relationship between designers and software developers. Or, even better, software developers should be made aware of design principles.

Norman closes out chapter one, of *The Design of Everyday Things*, by stating challenges in design include that “design requires the cooperative efforts of multiple disciplines” (The Design Challenge, para. 1) and that the hard part of design is “to convince people to understand the viewpoints of others” (The Design Challenge, para. 3). These points emphasize the key points of the first two chapters as they relate to human behavior. While the design of the devices people use is complex and much of human behavior occurs at a subconscious level, it is important for human-centered design (HCD) to exist. HCD allows our society to operate and encourages innovative solutions. While seemingly complex, it is not impossible with the aid of Normans design advice and psychological concepts: affordances, signifiers, constraints, feedback, mapping, and concept models (Norman, 2013, Chapter 1, Fundamental Principles of Interaction).

## References

Norman. (2013). *The Design of Everyday Things: Revised and Expanded Edition* (Rev. and expanded ed.). Basic Books.

Robinson, J. (2023, January 12) *RE: Balancing Software Security with Design Advice* [Discussion Post]. Drexel University Learn. https://learn.dcollege.net/