1 Introduction to User-Centered Design

In 2009, Jakob Nielsen and his user testing team at the Nielsen Norman Group reported findings from a study on mobile device usability. The results were disappointing. In fact, Nielsen (2009) wrote that watching users "suffering during our sessions reminded us of the very first usability studies we did with traditional websites in 1994. It was that bad" (par. 6).

Most disconcerting and surprising to Nielsen and his team, as well as mobile device designers everywhere, was that users carrying out tasks as part of the 2009 study performed worse than similar users carrying out the same tasks with mobile devices in 2000. In some cases, users of 2009 devices took one or more minutes longer to find something as simple as the local weather forecast than users of 2000 devices.

How could such a result be possible? Those who used mobile devices at the turn of the century remember all too well how frustrating using mobile devices could be, especially when trying to access information from the web. These devices featured tiny screens as well as a tiny touch-tone telephone keypad interface requiring you to choose a letter from a range represented on a number (see Figure 1.1 for a sample of a 2000-era mobile device). Added to difficult input of text was poor network connectivity and interfaces displayed on phone screens clearly designed to display on 1024 × 768 pixel desktop screens. Still, users in 2009 managed to do worse than users in 2000.

Obviously, mobile devices have improved since 2000. In fact, the iPhone was available in 2009 (introduced to the market in 2007) when Nielsen Norman reported on their study. The iPhone and other smart, touchscreen phones performed better in the study than other mobile devices. Nevertheless, despite their better features and quick, widespread adoption, they were not overnight panaceas. The iPhone in 2007, the iPad in 2010, and other extraordinary technology, built with an array of never-before-seen features, have not brought users solutions that deliver mistake-free experiences. One need only look at Google's Wave (Figure 1.2) platform to understand that building state-of-the-art technology does not guarantee a better user experience or market success.

Google Wave was a synchronous (or real-time) communication platform (as opposed to asynchronous communication such as email). The idea was that, rather than cobble together any variety of different technologies to manage projects, share files, instant message, talk, and email, a group of people working together or even just a group of friends or family could use Wave to do everything under one roof.

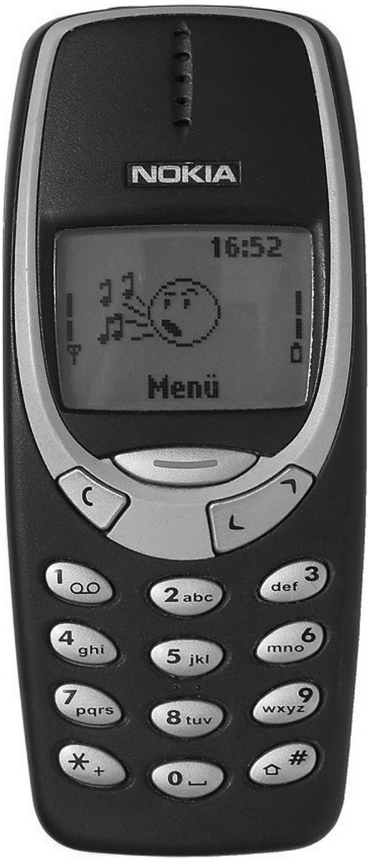

FIGURE 1.1 2000 era mobile device, the Nokia 3310 "Nokia 3310 blue R7309170 wp." (Rainer Knäpper, Free Art License; http://artlibre.org/licence/lal/en/.)

Wave was built to give users every tool they might possibly need to share information. One quick glance at the Wave interface certainly demonstrates this:

- Traditional email supplemented by photo profiles of everyone online.
- Built-in search function pulling up returns from key words shared in email or in the ongoing chat window in the upper right of the screen.
- An archive of previous communications, file storage, even the capability of playing back parts of previous meetings that had been recorded and stored. Google Wave had it all (Parr 2009).

With much fanfare and promise, Wave debuted to "an enthusiastic crowd of developers" in 2009 (Arrington 2010, par. 2). One year later, with only minimal adoption by users, development stopped. By 2012, Google had officially killed it.

At EyeGuide, the eye-tracking research and control company cofounded in 2011 by Brian Still and Nathan Jahnke, one of the development team's favorite sayings is

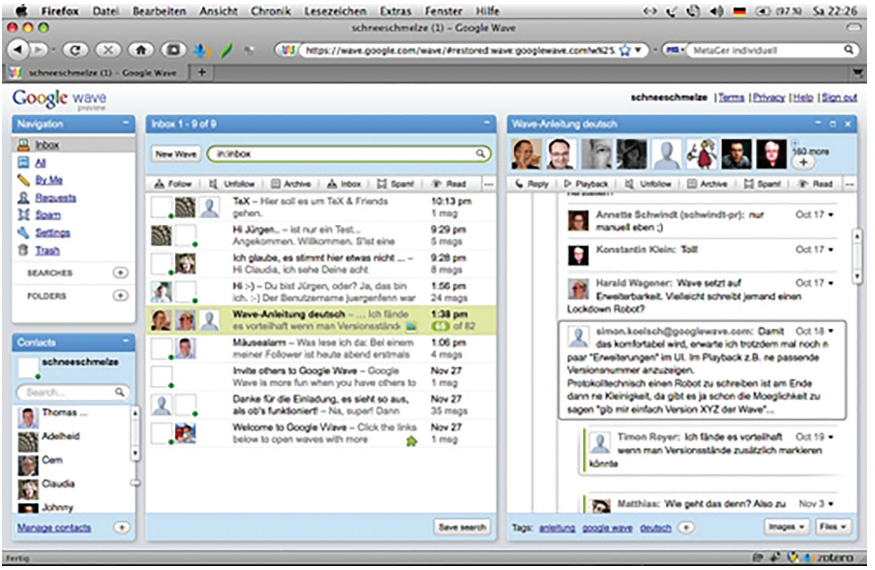

FIGURE 1.2 Google Wave "Google Wave in Firefox 3.5 under Mac OS X Leopard." (Jürgen Fenn. Taken on November 28, 2009, Creative Commons Attribution 2.0; https://creativecommons.org/licenses/by/2.0/.)

"making something awesome don't mean using something awesome." With all due respect to Kevin Costner's character in *Field of Dreams*, a fantasy baseball movie from the 1980s, people will not necessarily come if you build it. Just as users struggle with poor technology and don't want to use it, users also struggle with well-designed technology and don't want to use it. If both technologies are built poorly or built well, each has an equal chance of frustrating users or causing them to perform poorly. This results in not only a bad marketing outcome, but often a bad financial outcome as well.

In "Why Software Fails," Robert Charette (2005) writes that software "failures are universally unprejudiced: They happen in every country, to large companies and small; in commercial, nonprofit, and governmental organizations; and without regard to status or reputation. The business and societal costs of these failures…are now well into the billions of dollars a year" (43). One key reason they fail is poor communication between developers and users, which can be seen in the example of Oxford Health Plans. In 1997, Oxford employed a new automated business system without getting feedback on its development or from its users—patients and caregivers. As the business expanded, the system couldn't keep up, resulting in $400 million in uncollected payments and $650 million more in unpaid bills. When public investors learned of Oxford's problems, its stock dropped from $68 dollars a share to $26, a $3.4 billion dollar loss in one day (Parr 2009, 45). Managers and developers initially made no effort to build a system that accommodated user needs, and even once the system was operational, their continued lack of effort to gather feedback from users about how to correct the system ultimately caused it to fail.

Of course, not every company is like Oxford. Charette (2005) mentions Praxis High Integrity Systems as an example of a company that requires "customers be committed…not only financially, but as active participants in" a new system's creation (48). Praxis, according to Charette, focuses a "tremendous amount of time understanding and defining the customer's requirements, and it challenges customers to explain what they want and why. Before a single line of code is written, both the customer and Praxis agree on what is desired, what is feasible, and what risks are involved, given the available resources" (49).

Other organizations are also leading the way in creating development processes and products that engage, proactively, their intended users. Pinterest, Uber, and Airbnb, among others, are examples of organizations that don't focus on making things for users, but rather, as Patrick Stewart (2014) writes, "facilitate the exchange of value between users" (par. 2). At the core of what they do is something Stewart refers to as design thinking.

Design thinking is iterative in nature, but at the heart of how it works is empathy. Designers need to have empathy for a user's experience. As Stewart (2014) notes, Airbnb founder Brian Chesky "rents Airbnb apartments while hosting other users at his apartment" (par. 20) to better "empathize with both renters and hosts and experience the quality of consumer-producer interactions first hand." From this participation, he learns, in an experiential fashion, what customers must do to find a place to stay, what obstacles get in the way of that effort, what tools they use, how and when they engage with those tools, and how those tools do and don't work to aid users in accomplishing their goals. He doesn't let his users design Airbnb for him, but he doesn't design the application without understanding their motives and needs.

Unfortunately, for every Airbnb, there are as many or more that do not subscribe to user-centered design (UCD) thinking. The OpenOffice computer mouse prototype (Figure 1.3), unveiled in 2009, is an example.

Complete with 18 features, including an Analog Xbox 360-style joystick with optional 4-, 8-, and 16-key command modes, 512 kb of its own flash memory, and three different button modes, the OpenOffice mouse represented in form and function the most obvious violation of Steve Krug's (2006) simple but effective first law of usability: "Don't make me think!" (11). Krug explains, "It's the overriding principle—the ultimate tie breaker when deciding whether something works or doesn't…" (11). Krug writes primarily about web design, but this sentiment is true for any product or process people use. To use a product effectively, "it should be self-evident. Obvious" (11).

This truth is not limited to examples of over-designed products or processes. It is easy to pick on things, like the OpenOffice mouse, that fail to be understandable and are not usable for their intended users because they have too many features or functions to comprehend. Certainly, too much utility, or the intrinsic capability of a product or process to complete certain functions or tasks, in a product is a factor in whether users can figure out how to work it or how to use it in particular situations. High-utility interfaces cannot always be avoided. Aircraft cockpit controls, which have been studied since World War II in an effort to make them intuitive to pilots, are an example of high-utility interfaces that are necessary.

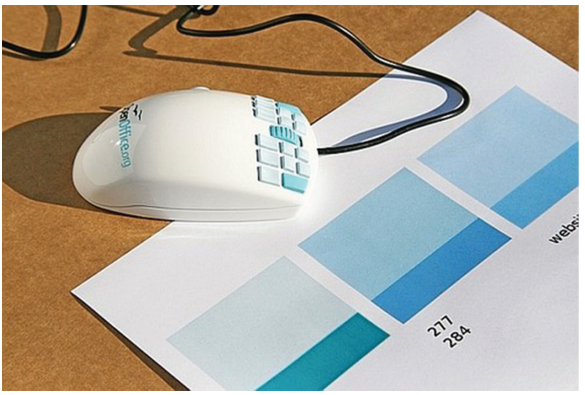

FIGURE 1.3 OpenOffice mouse with 18 built-in functions. (Reprinted with permission from UXcertification.com.)

There are equally unusable low utility interfaces or interfaces that you can only do certain obvious things with while using them. The elevator control images in Figure 1.4a and b illustrate that it is quite possible to make what should be a user-friendly interface unusable. In Figure 1.4a on the left, the elevator control seems simple and straightforward, but the buttons for the floor increase up and to the right in an awkward pattern; Floor 16's button is actually below Floor 5's button. In Figure 1.4b on the right, the floor buttons increase from left to right and include an icon, BR, whose physical location in the building is unclear.

Take a walk around your house (or office or campus) and you're likely to find something designed poorly. In our everyday lives, there are countless examples of things made for us to use but not focused on how we would use them in our real-world experiences. For example, Brian was having lunch at an upscale restaurant. The sink faucets in the bathroom were automatic although Brian initially thought they were broken. Nothing on the faucet itself told Brian how to use it. The soap dispenser made sense, but even when Brian waved his hands beneath the faucet, that action didn't trigger the water to run. In desperation, he used his soapy hands to push down on the faucet, which he noticed was a little taller than it should be. His hunch paid off. The water started. Cold water? Hot water? He couldn't control what he got. Luckily, the temperature wasn't too extreme and he could wash off the soap. However, the water didn't shut off. Brian looked around in a panic. Did he just break the faucet? He reached across and pushed down the faucet in the sink to his right and it came on and started the flow of water. That told Brian the faucet worked by pushing. Eventually, the water stopped, but there were many anxious moments about what might have happened had the water not shut off.

Beyond Brian's mundane example, there are examples of where poor design leads to or exacerbates tragic accidents. The Bhopal, India, gas leak in 1984 and the Chernobyl, Ukraine, nuclear reactor disaster in 1986 are both (Meshkati 1991) prime examples of failures in systems because of a noticeable lack of consideration for human factors. At Chernobyl, despite some engineers' heroic efforts to stop the nuclear reactor's dangerous meltdown, there were no systems in place to allow

(a)

(b)

FIGURE 1.4 Images of confusing elevator controls. (a) Awkward pattern for organizing elevator buttons and (b) what does BR mean? Another example of confusing elevator button labeling. (Reprinted with permission from John Bartholdi, http://www2.isye.gatech.edu/~jjb/misc/elevators/elevators.html.)

operators to carry out actions in emergencies to control the reactor. There weren't alarms to warn them of danger, and by the time the reactor began to fail, there weren't sufficient warnings or tools to prevent the unfolding disaster. To this day, the region around Chernobyl is uninhabitable, and more than 7,000 reported cases of thyroid cancer in children under age 18 have been attributed to the radiation fallout (Nuclear Energy Institute 2015).

Union Carbide's storage plant leaked dangerous gas into Bhopal, India, in December 1984. Almost 3,800 people were killed and another 200,000 injured. Again, although numerous physical and organizational failures led to this disaster, poor design not centered on the needs and capabilities of the plant's operators contributed to the disastrous outcome. Meshkati (1991) notes that plant staff and workers, undertrained and overworked, were responsible for monitoring too many gauges, some of which were often broken. Ironically, many signs informing responsible staff about monitoring plant safety were written in English although the staff predominantly spoke and read Hindi.

In *Set Phasers on Stun*, Casey (1998) catalogues a distressing number of tragic stories stemming from mistakes caused by how users interacted with products. They include the death of a cancer patient repeatedly given the wrong dosage of radiation because the technician had inadvertently typed in an X instead of an E to operate the radiation therapy machine. The crash of an Air France Airbus in 1988, which killed two children and injured many others, is another example. After this crash, and a subsequent one with more fatalities in 1990, Airbus concluded that pilots were too overconfident with how well the automatic landing control system worked in the aircraft. As such, pilots were not sufficiently trained to interpret the system or how to act if the system was providing incorrect data (106).

Endsley and Jones (2004) argue that one reason there is such high operator error in aircraft accidents, for instance, is "not the result of [a] faulty human, but rather the direct result of technology-centered designs that have been ill-suited for promoting high levels of sustained human performance over the wide range of conditions and circumstances called for in the real world. Rather than human error, a more accurate term would be design-induced error" (6). These are examples of technology-centered designers who build the baseball fields and then expect people to come. It's not that users aren't considered in the technology-centered design (TCD) process; rather, they are one of many factors. The primary focus in TCD is to create technology capable of fulfilling certain functional objectives. Only at a later stage of the process, after design and testing for errors in code, are users in the TCD model considered. Unfortunately, then, this feedback comes too late to be constructive. Instead, only surface or cosmetic errors in design can be addressed.

This TCD approach is often called the waterfall approach (Figure 1.5). The product or process being designed falls down toward the users like a waterfall, and each phase of the process is started and completed in succession. The problem with a waterfall is that a lot of water is falling down precipitously, all toward the users, and not any of the water can go back from where it came. This process doesn't allow for much iteration or empathy.

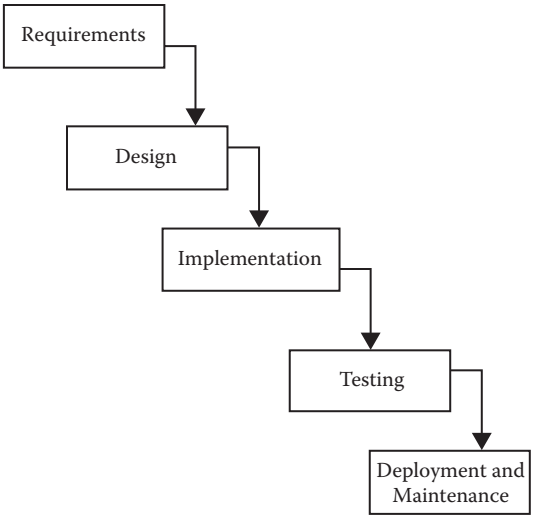

FIGURE 1.5 The waterfall, TCD approach.

The TCD designer needs to answer certain questions:

- What's the deadline?
- What features does the product require?
- What materials are available for constructing the product?
- What's the budget?
- What does the designer believe will make the product successful, aesthetically, functionally, and financially?

TCD designers may try to get a sense of who will use the product through feedback from surveys or focus groups in the beginning, or at the end—when the product is nearly ready to go to the market—when a select group of beta users may give their feedback about the product. However, the actual users and their experiences don't show up anywhere during the design process that makes a product usable for them.

Brian's company, EyeGuide, ironically, built a product born from the classic TCD waterfall approach. In 2012, EyeGuide debuted to the world the first truly affordable assistive technology called EyeGuide Assist. Over 23 million Americans, not to mention millions more elsewhere, have limited or no hand functionality; Brian and his team aspired to rectify this with EyeGuide Assist, which allowed users to control the computer cursor with their eyes. To type or click, Assist enabled users to blink to type or click, to make a sound such as "type" or "click," or to stare at or dwell over a letter on a keyboard or a link on the screen (Figures 1.6 and 1.7).

What EyeGuide promised with all this functionality was also affordability. Many other assistive tools on the market in 2012, especially those that made use of eye-tracking technology, were prohibitively expensive, especially for people with assistive needs on fixed incomes. Having already figured out how to engineer effective,

FIGURE 1.6 EyeGuide Assist in action. (Courtesy of EyeGuide.)

FIGURE 1.7 EyeGuide Assist. (Courtesy of EyeGuide.)

affordable eye tracking with a previous product, EyeGuide felt that Assist offered the right combination of price and functionality.

Debuting at the January 2012 Assistive Technology Industry Association (ATIA) conference in Orlando, Florida, Assist stunned many of those in attendance, including other competitors and disability advocates. It was affordable and could be used to type, even paint, with the eyes. The subsequent press coverage was strong, and EyeGuide launched Assist into the marketplace, taking and fulfilling dozens of orders from all over the world.

It was toward the start of the second quarter in April 2012 that complaints and requests for refunds from Assist customers began to trickle in. Brian and his team

worked overtime trying to understand the problems and come up with solutions. Everyone they trained in person still had a unit. One of their more high-profile customers, a former NFL football player diagnosed with amyotrophic lateral sclerosis (ALS), even appeared on an episode of HBO's *Real Sports* with Bryant Gumbel using Assist (Figure 1.8). Still, the refund requests continued, and eventually more than 50% of all users wanted their money back.

Short videos on how to use Assist, increased live and virtual support, and other support tools were implemented to remedy the decreasing sales and increasing refund requests; however, in late 2012, EyeGuide ultimately realized that no amount of support could correct the mistakes that had been designed into Assist, and the product was discontinued. In order to be affordable, Assist needed to cut corners on certain functionality. As Figure 1.8 shows, users or their caregivers had to adjust the camera to fit over the eye, and the LED arm had to be adjusted independently so that the eye could be found in the software and calibrated accurately. Otherwise, the computer mouse, once under Assist's control, wouldn't behave properly. Without head movement compensation built in, users also couldn't move their heads without the risk of losing calibration. Although these limitations were acceptable for researchers carrying out short eye-tracking tests of what users looked at when, for example, buying something online, the ability to put the system on, calibrate, and

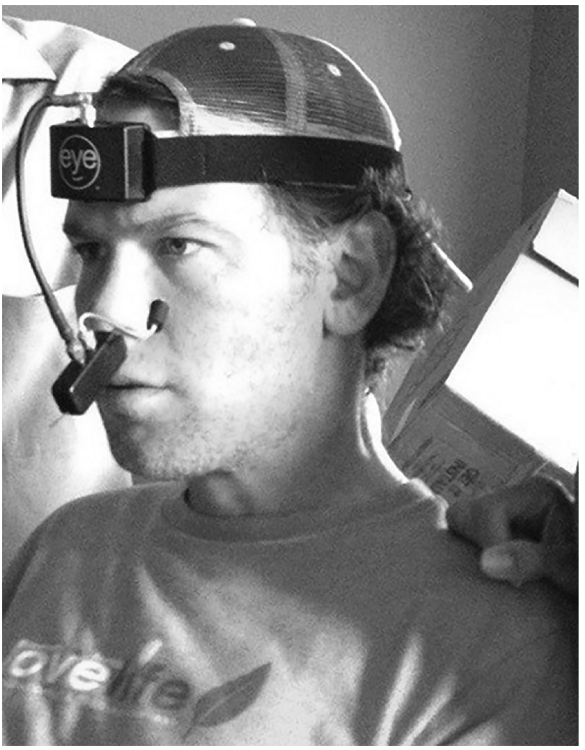

FIGURE 1.8 Former NFL player with ALS using EyeGuide Assist. (Courtesy of EyeGuide.)

quickly function without much additional work for a sustained period of time were all affordances required by assistive users.

No product is cheap enough if it does not work for its intended users. EyeGuide made assumptions about Assist's target user population that were wrong, and only after the product went to market did feedback from users become a part of the process for understanding what they needed and wanted. Technology-centered designers want their products to work. EyeGuide designers wanted to be successful for its investors and for its users. They wanted to make effective, affordable assistive technologies. However, they never bothered to understand what users needed or wanted or how they would use the technology in their everyday lives. They didn't design, in other words, for the user's experience.

Marc Hassenzahl (2014) calls this a tremendous challenge for designers. As the failures of EyeGuide Assist or Google Wave show us, "user experience is not about good industrial design, multi-touch, or fancy interfaces. It is about transcending the material. It is about creating an experience through a device" (Hassenzahl 2014, sec. 1, par. 4). Whereas "the traditional instructional systems design…approach," Baek et al. (2008) write, "is reductionist in nature and that it tends to solve a problem by fragmentation, one stage at a time" (660), design for the user experience is more "subjective, holistic, situated, dynamic, and worthwhile" (Hassenzahl 2014, sec. 3, par. 3). The user experience approach starts with why (Hassenzahl, 2014). The designer does not ask, can I make this or how can I make this? Rather, the designer first asks, why am I going to make this product, and why is it required to assist users in fulfilling an experience?

Donald Norman (2014), responding to Hassenzahl's (2014) emphasis on making design about experience, writes, "Design has moved from its origins of making things look attractive (styling), to making things that fulfill true needs in an effective understandable way (design studies and interactive studies) to the enabling of experiences" (sec. 8, par. 6). It is Norman, writing in 1986, who first proposes how to accomplish this as a means to bridge a natural gap that occurs whenever a designer makes something for a user. According to Norman (1986), a user has personal goals for acting, such as taking money out of an ATM or ordering a shirt from an online store. The problem for designers is that users express these goals in psychological terms. Users want to use products that allow them to easily, effectively, and successfully perform their tasks. However, the system made for them is often problematic because its "mechanisms and states are expressed," according to Norman (1986), "in terms relative to it" (38), not necessarily the user.

Norman (1986) calls these opposed states the "gulfs of execution and evaluation" (38). The user has goals and must interact with the system to execute these goals. The system has a particular interface that needs evaluation. Execution or intention flows from user to system; evaluation or interface, and the means to interpret it, flow from system to user. To put this another way, there is a gap when a user is asked to accomplish a goal with a product. In a perfect world, you would just wish for money and get $40 in your pocket; there would be no need to interface with anything as your wish would be realized by your psychological goals. Unfortunately, this really isn't possible. Therefore, if you're a designer, it is incumbent on you to figure out how to make a product that is as similar and realistic as possible to help the user accomplish goals without too much interpretation or thinking (or at least not any more thinking than is necessary). Obviously, some

products, like flying an Airbus aircraft, demand more complicated tasks. Nevertheless, Hassenzahl's (2014) and Krug's (2006) arguments still hold true. To bridge the gap for a better user experience, meaning users can accomplish their goals in a satisfying, effective way, you cannot design from a perspective that emphasizes what the system has to offer and expect the user to adapt to the product. Rather, you should design the system to adapt to the user's psychological, situational, emotional, and intellectual needs.

Therefore, Norman (1986) posits that the best approach to helping designers and users, one that most effectively bridges the gap between evaluation and execution, is through UCD: "User-centered design emphasizes that the purpose of the system is to serve the user, not to use a specific technology, not to be an elegant piece of programming. The needs of the users should dominate the design of the interface, and the needs of the interface should dominate the design of the rest of the system" (61).

Now UCD designers don't stop being designers or dealing with deadlines, budgets, and other constraints just because they focus the design process on the user. They still follow best design practices and adhere to the same requirements found in the waterfall approach. All the waterfall questions matter to them as well. But, and this is key, other questions also matter:

- How would users use this product?
- How do they think about it or similar products?
- How do they interact with this product?
- Where do they interact with it and under what circumstances?

The UCD designer observes and interviews users, evaluates the market and competitors, and uses all of that feedback to determine the requirements needed in a product before it is even prototyped. Instead of waiting until the end of the process for users to validate a nearly completed product, UCD designers consult with representative users throughout the process. Users test out multiple prototypes, and designers make revisions to the design each time. Eventually, the product is molded to the users' needs, desires, and situations.

It doesn't end there either. UCD designers also continue to consult users even after the product is released in order to make improvements and spin off new iterations or products. Many of the applications we use on a daily basis incorporate this attitude of continual design. In fact, application designers often embed protocols into their software that record patterns of use, and they regularly consult user reviews and recruit users for testing in order to develop better versions of their apps.

UCD *doesn't* mean that users are in control of the design process, have the last word on a big decision, or make their opinions count more than others. You also can't make something unique for every individual. Rather, necessary elements of UCD include knowing several key things:

- How the user thinks and interacts with a product
- What the user requires
- What others have done before, both bad and good, to engage users
- What the company wants to accomplish with the product it's making for the intended users

Instead of users evaluating the product at the end of the chain or only evaluating more developed prototypes, UCD values user engagement from the very beginning of the design process. It takes advantage of a variety of methods and techniques that are incorporated into an approach that allows product designers to think about users' needs and visualize how they will use a product from the earliest stages, even before something has been sketched out on paper. If you make things for people to use, you must understand and design for how people will use what you make. Lowgren and Stolterman (2004) have called this "thoughtful design" (5). Basically, thoughtful design "refers to the process that is arranged within existing resource constraints to create, shape, and decide all use-oriented qualities (structural, functional, ethical, and aesthetic) of a digital artifact for one or many clients" (5).

"A key premise of user-centered design," write Endsley and Jones (2004), "is that while a person does not need to perform every task, the person does need to be in control of managing what the systems are doing in order to maintain the situation awareness needed for successful performance across a wide variety of conditions and situations" (11). Essentially, good UCD is not about giving users what they want or making decisions for them. Rather, it is giving them enough control to understand and manage the system in multiple types of situations.

While researching a new application for law enforcement, Brian rode along with a state trooper. He noticed that, on different occasions, particular sounds would prompt the trooper, even in the middle of other tasks, to look at the touch screen monitor positioned to the right of his steering wheel. After the ride-along, Brian asked the trooper about the sounds and their meanings. When everything was in a normal state, there were no unnecessary sounds; thus, the trooper was not distracted and could pay attention to driving his car and viewing other vehicles on the road. However, when a fast beep sound occurred, the trooper quickly looked down at the monitor and noted the position of a flashing dot on a map. He then went back to driving. This beeping sound and corresponding dot indicated that another law enforcement officer had pulled over a vehicle. Should a louder, deeper, longer sound occur, it would notify the trooper that the other law enforcement officer needed assistance. In other words, the sound signaled a change in the situation and because the trooper already made a mental note of the location where the other officer had stopped the vehicle, the trooper could react faster to provide assistance. Further, the loud, deep sound was never used for another purpose. Reserved just for an officer in distress, the trooper would never be confused by its meaning. The interface of the system was designed to aid the user in having control over different situations.

Good UCD is understanding users, designing a product, and doing whatever it takes to make that product work them. Baek et al. (2008) points out that user participation can be weak or strong, in terms of interaction, length, scope, and control. Sometimes users are present at the beginning of the design process, providing constant feedback, trying out prototypes, even contributing to designs. This is strong participation. In other situations, the product's representative users may be recruited to test out prototypes as they are finished, providing more controlled feedback for certain times in the process. This is a weaker form of participation. No one approach,

weak, strong, or a combination thereof, is necessarily better or worse than another as long as the participation fulfills project goals and users are involved in some way in the design of the product they will ultimately use.

Along those lines, UCD is a "big tent" approach that draws from a multitude of techniques, methods, and other design concepts. Sometimes you will hear people refer to UCD as more or less another way to describe usability testing. However, they aren't synonymous. We certainly need to administer usability testing of a product as part of a UCD process, but it is just one method that allows you to focus the design process on users. To carry out effective UCD, you may need to understand and apply knowledge of the following:

- Information architecture
- Ergonomics
- Market acceptance
- User preference
- Functionality/utility
- Coding
- Aesthetics

As Pratt and Nunes (2012) write, "well-designed information" has to account for many things (16). One need only watch people try to open doors with unfamiliar handles to understand the importance of everything working well and centering on users' needs. While traveling for work, Brian watched several people get confused by an automated revolving door used to exit the terminal at an airport. There was no guard to warn users that leaving through this exit meant exiting the secured area of the airport. Other than a small sign above the door that read "No Re-Entry," there was no other cue to signal that if users left through the revolving exit door they couldn't come back. Nevertheless, a number of users exited and then attempted to reenter. Likely they were following the crowd of other users who did want to exit and get their baggage. At least one of them was on the phone. Another had her head down and appeared tired.

Should they have seen the sign above the door and acted accordingly? Perhaps, but should the sign or another visual cue have been better placed, given the situation, to guide them in making the correct decision? You design for real people, in the real world, so what you make should reflect an awareness of that. According to Pratt and Nunes (2012), "Understanding who we design for, what they want and need, and the environment in which they will use our designs, is not only a good way to guarantee a successful product, but also a safer, saner world" (16).

UCD is about the entire process from start to finish. For example, if you are a developer, you can follow every protocol to ensure that the code you've implemented has zero defects, yet the product the code powers could still fail because the product wasn't centered adequately on the user's needs. This is also true for aesthetics. Beautiful creations are often rejected because users don't know how to use them, and more than once users have indicated they liked something or preferred that it be made a certain way but didn't buy or use it when it came to the market.

The opposite is also true. A large community of Apple users scoffed at the idea of the iPod, wondering why anyone would want to download individual songs on such a simple device. But Steve Jobs, and Apple, understood that simple design is often the best approach to meeting users' needs. The iPod also granted control directly to the users. The users could control what they downloaded and played. Millions of iPods later, Apple was right, just as it was about the Apple Store, the iPhone, and the iPad. Rather than throw hundreds of things at users in a confusing interface, Apple simplified design and maximized technology, such as touch control, a larger screen for viewing, and a larger keyboard for typing, so that certain affordances were included. They also limited the number of required applications, allowing users to control what was added to their device.

UCD is not focused on just particular aspects of a product's development; instead, it is a thoughtful, comprehensive view of the entire process that puts users' needs front and center. Users' needs are not the only determiner, but they do govern decisions made about aesthetics, architecture, ergonomics, etc. You want these elements of your design done correctly but not in isolation. You design for users who live, work, and play in a real world. UCD focuses on users using things in that world.

The issue then becomes, how do we do it? If we understand what UCD is, how do we implement it in a repeatable, effective way so that it can be successful for you and users, over and over again? This question is what this book attempts to answer. It's important to understand the big picture. We provide that knowledge in this chapter and the next two chapters that look at the history behind UCD and its fundamental governing principles. The bulk of this book, however, aims to give you a comprehensive overview of how to discover, design, and deliver effective UCD.

So let's get started. As Jesse James Garrett (2011) writes, "The concept of user-centered design is very simple: Take the user into account every step of the way as you develop your project" (17). Of course, doing this and saying it are two very different things because "the implications of this simple concept," Garrett continues, "…are surprisingly complex" (17). How should we involve users effectively? Why should users be involved? Are there other ways to design that don't focus so intently on users, and how is UCD different? What events or other factors compelled the creation of UCD as a new approach to design? In this next chapter, we'll focus most on answering this last question as we explore the origins of UCD.

TAKEAWAYS

1. Building state-of-the-art technology does not guarantee a better user experience or market success. Technology built poorly or built well has an equal chance of frustrating users or causing them to perform poorly.
2. Poor communication between developers and users about why, how, and where users use things is a key reason why technologies fail.
3. UCD accommodates user engagement from the very beginning of the design process. Getting feedback or buy-in from users before a product is designed is critical.
4. Design thinking means facilitating the exchange of value between users and having empathy for the user experience.

5. Too much utility or too many features impacts whether users can (or want to) figure out how to use a product.
6. Poor design leads to poor business results and potentially dangerous consequences.
7. Technology-centered design favors fulfilling functional rather than user objectives.
8. No product is cheap enough if it doesn't work for its intended users.
9. A gulf between execution and evaluation means that user goals do not align with how the system evaluates and interprets user input to operate.
10. We should design to make the system adapt to a user's psychological, situational, emotional, and intellectual needs. This trumps a design perspective that emphasizes what the system has to offer above all else and expects users to figure it out.
11. UCD considers a user's mental model: how users might use a product, how they think about it, and how, where, and under what circumstances they interact with it.
12. UCD is not about giving users what they want or making decisions for them. It's about giving them enough control to be able to understand and manage the system they are using, dependent on the situation.

DISCUSSION QUESTIONS

1. Why might a technology, process, or product be more difficult to use now than in the past?
2. Think about EyeGuide's euphemism, "Making something awesome don't mean using something awesome." What's an "awesome" technology or product you have used that turned out not to be so awesome?
3. Envision yourself as a product developer for a new mobile app. What questions might you ask a user before you begin the design process? What types of information do you want to know about them?
4. Think of a technology that you use—by choice or not—on a regular basis. Is it easy to use? If so, why? If it's difficult to use, what would make it easier to use? Discuss how these ease-of-use considerations might be a result of the method, location, or time you use the product.
5. Review your campus or workplace emergency plan for a fire, tornado, or active shooter situation. Are the plans clear and easy to understand and execute? What human factor considerations might help improve user understanding of the procedures? Consider sharing the results with your school or workplace.
6. Think of a step-by-step process that you know how to complete (e.g., getting dressed, cooking, shooting a basketball). How difficult would it be to realize that you made an error and not be able to change anything until the process was complete (waterfall approach)? At what stage would it be easiest to make changes?

REFERENCES

Arrington, M. "Wave Goodbye to Google Wave." *TechCrunch.* August 4, 2010, accessed February 18, 2016. http://techcrunch.com/2010/08/04/wave-goodbye-to-google-wave/.

Baek, E., K. Cagiltay, E. Boling, and T. Frick. "User-Centered Design and Development." In *Handbook of Research on Educational Communications and Technology*, edited by M. Spector, D. Merrill, J. Van Merrienboer, and M. Driscoll, 659–670. New York: Lawrence Earlbaum Associates, 2008.

Casey, S. *Set Phasers on Stun: And Other True Tales of Design, Technology, and Human Error*. Santa Barbara: Aegean, 1998.

Charette, R. "Why Software Fails." *IEEE Spectrum* 42, no. 9 (2005): 42–49.

Endsley, M. and D. Jones. *Designing for Situation Awareness: An Approach to User-Centered Design*. New York: CRC Press, 2004.

Garrett, J. *The Elements of User Experience: User-Centered Design for the Web and Beyond*. Berkeley: New Riders Publishing, 2011.

Hassenzahl, M. "User Experience and Experience Design." In *The Encyclopedia of Human-Computer Interaction*, 2nd ed., edited by M. Soegaard and R. Dam, sec. 3. Interaction Design Foundation, 2014. https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/user-experience-and-experience-design.

Krug, S. *Don't Make Me Think*. Berkeley: New Riders Publishing, 2006.

Lowgren, J. and E. Stolterman. *Thoughtful Interaction Design*. Cambridge: MIT Press, 2004.

Meshkati, N. "Human Factors in Large-Scale Technological Systems' Accidents: Three Mile Island, Bhopal, Chernobyl." *Industrial Crisis Quarterly* 5 (1991): 131–154.

Nielsen, J. "Mobile Usability: First Findings." *Nielsen Norman Group*. July 20, 2009, accessed February 18, 2016. https://www.nngroup.com/articles/mobile-usability-first-findings/.

Norman, D. "Cognitive Engineering." In *User-Centered System Design: New Perspectives on Human-Computer Interaction*, edited by D. Norman and S. Draper, 31–61. Hillsdale: Lawrence Erlbaum Associates, 1986.

———. "Commentary by Donald A. Norman." In *The Encyclopedia of Human-Computer Interaction*, 2nd ed., edited by M. Soegaard and R. Dam, sec. 3.8. The Interaction Design Foundation, 2014. https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/user-experience-and-experience-design.

Nuclear Energy Institute. "Chernobyl Accident and Its Consequences." *Nuclear Energy Institute*. March 2015, accessed February 18, 2016. http://www.nei.org/master-document-folder/backgrounders/fact-sheets/chernobyl-accident-and-its-consequences.

Parr, B. "Google Wave: A Complete Guide." *Mashable.* May 29, 2009, accessed February 18, 2016. http://mashable.com/2009/05/28/google-wave-guide/#ZmTD5nnbhGqT.

Pratt, A. and J. Nunes. *Interactive Design: An Introduction to the Theory and Application of User-Centered Design*. Beverly: Rockport, 2012.

Stewart, P. "How to Design a Billion Dollar Company." *Applico.* May 15, 2014, accessed February 18, 2016. http://www.applicoinc.com/blog/how-to-design-a-billion-dollar-company/.

Taylor & Francis

Taylor & Francis Group

http://taylorandfrancis.com