

# **Human-Computer Interface Design and Usability Testing**

**Audrey Troutt and Daniel Sheiner**

**CIT 595, Spring 2007**

## Abstract

Usability is an important topic for software developers, yet even now, despite over thirty years of research and discussion on human-computer interaction, good principles for hardware and software interface design are often completely ignored. Usability becomes a mere buzzword while real usability research is neglected despite the fact that good interface design leads to greater commercial success. We will examine the history of usability work, specifically in software development, and present the findings of scholars who have proven the value of usability work in software development.

In addition, we will describe the basic principles by which usability experts evaluate and improve the human-computer interface. A good interface does not require the user to think too much, provides for diverse users, mitigates human error and makes users feel good. To determine whether a program meets these objectives, scholars use both laboratory and field observations and produce written or oral reports. We will elaborate on what the scholars look for and how their findings are integrated back in the design process.

Prevailing computer interface design practices have remained consistent for the last 20 years. Some scholars say that is about to change, that we can expect a revolution in HCI, just like the one that brought about the personal computer in the 1970's. For this reason we will give brief descriptions of emerging technologies which offer new ways of interacting with computers, like Jeff Han's multi-touch screen, as well as new software that has been designed with usability as a top priority.

## Table of Contents

<u>Section</u>	<u>Page</u>
I. Introduction: History and Definition of Usability	2
II. Design principles	5
III. Usability Testing	11
IV. Conclusion: the Future of Human-Computer Interface Design and Usability	16
References	18
Appendix: Task Summary	19

## I. Introduction

Usability is one of many factors that software designers must take into account when designing a new user interface. Everyone has seen examples of frustrating, baffling, or simply poorly-designed user interfaces; in these cases it may be obvious what is wrong with the design and what one might do to fix it, but what if the usability problems are not so obvious? What separates a merely acceptable interface design from a really excellent design? These questions were the first motivation for this exploration of usability and human-computer interface design.

As it turns out, usability, also known as human factors, user-centered design, and human-computer interface design, has become a major field in both academia and industry.<sup>1</sup> This has to do with the increasing role computers play in our lives--the more people need to interact with computers, the more they are going to demand well-designed interfaces. Software developers are wise to answer to this demand; there are commercial advantages to designing better interfaces, as demonstrated by the experiences of such companies as Apple, Eastman Kodak, IBM, Microsoft, Rank Xerox, and SAAB.<sup>2</sup> Well-designed software can increase productivity, reduce human error and make software more enjoyable to use.<sup>3</sup> However, usability is sometimes neglected not because it is considered unbeneficial, but because it is thought to be too expensive, which may be a misconception.<sup>4</sup> In other cases, software developers assume usability is little more than common sense and as such does not justify the cost of usability testing and evaluation.<sup>5</sup>

There can, however, be even greater costs for ignoring usability factors in interface design, as proven by dramatic examples of fatal accidents resulting from interfaces' failures to effectively display critical information. In 1987, "An American aircraft crashed on take-off at Detroit killing 156 people. The accident was attributed to pilot failure to recognize that the flaps were in the wrong position for take-off."<sup>6</sup> A skilled interface designer would know to organize the interface in such a way that important information cannot be missed. In 1988 an Iranian commercial airliner was accidentally shot down by a Navy ship in the Persian Gulf. The crew misunderstood information from the ship's sensors and thought the plane was descending when it

---

<sup>1</sup> John Canny, "The Future of Human-Computer Interaction." *ACM Queue* 4. 6 (Jul.-Aug. 2006): 25.

<sup>2</sup> Paul Traub, "Optimising human factors integration in system design." *Engineering Management Journal* 6. 2 (Apr. 1996): 94.

<sup>3</sup> Robert E Filman, "Interface Pains." *IEEE Internet Computing* 8. 5 (Sept.-Oct. 2004): 5-6.

<sup>4</sup> Jakob Nielsen and Bill Curtis. "Applying discount Usability Engineering." *IEEE Software* 12. 1 (Jan. 1995): 98.

<sup>5</sup> Paul Traub, "Optimising human factors integration in system design.", 95.

<sup>6</sup> Ibid.

was actually ascending.<sup>7</sup> Thinking they were under attack, the crew shot down the plane, killing all 209 civilians on board. This tragedy might have been avoided if critical information about the location and movement of the plane hadn't been presented on distant consoles.<sup>8</sup> Other dangers may not be fatal, but financial. Paul Traub reports that in the UK, "as many as one-fifth of all software projects costing in excess of £1/2 million have resulted in systems which are inappropriate or unusable."<sup>9</sup> The costs and benefits of serious usability testing vary depending on the application of the interface, but thoroughly incorporating usability factors into the design of an interface improves the final product.

### History and Definition of Usability Work and Human-Computer Interface Design

Usability may generally be defined as "the extent to which a product can be used by specified users to achieve specific goals with effectiveness, efficiency, and satisfaction in a specified context of use."<sup>10</sup> This definition has evolved as the field of human-computer interface design has grown. When computers first came into wide use, the definition of usability was simple: "...the maxim 'easy to use,' which means requiring the minimum cognitive and physical effort, was the guiding objective for evaluation."<sup>11</sup> In general, the objectives of usability design are "to enhance the effectiveness and efficiency with which work and other activities are carried out," and "to enhance certain desirable human values, including improved safety, reduced fatigue and stress, increased comfort, greater user acceptance and increased job satisfaction."<sup>12</sup> Over time this definition has changed to reflect the multidisciplinary nature of usability work. Today, usability involves an understanding of psychology, physiology, anatomy, engineering, design, and management,<sup>13</sup> and as such, usability touches on many different aspects of the user's experience with an interface.

---

<sup>7</sup> Les Aspin, "Witness to Iran Flight 655," *New York Times*, November 18, 1988, national edition, <http://www.nytimes.com> (accessed April 15, 2007).

<sup>8</sup> Paul Traub, "Optimising human factors integration in system design," 95.

<sup>9</sup> *Ibid.*, 94.

<sup>10</sup> J. Karat and C. M. Karat, "The Evolution of User-Centered Focus in the Human Computer Interaction Field," *IBM Systems Journal* 42. 4 (2003): 535.

<sup>11</sup> *Ibid.*, 534

<sup>12</sup> Paul Traub, "Optimising human factors integration in system design," 94.

<sup>13</sup> *Ibid.*

Some definitions mentioned above share common ideas like efficiency and satisfaction, but continuing variation in the definition of usability is indicative of the ongoing development of our understanding of human-computer interfaces. One early pioneer was Alan Kay, who began work at Xerox Palo Alto Research Center (PARC) in 1970. He is credited with envisioning the idea of computers for everyday use.<sup>14</sup> Another Xerox PARC researcher, Don Massaro, introduced the idea of a WIMP (windows, icons, mouse, pointer) interface in 1976. Almost all computers we use today provide a WIMP interface. The first computer with this interface design was known as the Star, built by David Little.<sup>15</sup> What was interesting about the Star was not only the interface design, but also the methods the developers used to create it.

As we will discuss later, early and frequent user testing is critical to good user interface design. Understanding this, David Little and his collaborators at PARC used methods such as task analysis, user conceptual models, and rapid prototyping.<sup>16</sup> These are some of the same methods that usability specialists are championing today. The need to continually extol the virtue of these methods to software developers demonstrates that these testing and development methods are only sporadically applied in modern software design processes, perhaps because it took a long time to cultivate a community of usability specialists and a body of research to support it.

It wasn't until 1982 that the first conference on computer usability took place in Gaithersburg, Maryland. The conference resulted in the formation of the first Association for Computing Machinery Special Interest Group on Computer-Human Interaction (ACM SIGCHI).<sup>17</sup> Three years later, a committee of the Human Factors and Ergonomics Society set out to publish the first standards for human-computer interface design.<sup>18</sup> The project took over 15 years to complete, but in 1998 the International Organization for Standardization published ISO 9421, a standard for visual displays for office work. Subsequently the American National Standards Institute produced a standard for the ergonomics of human-system interaction.<sup>19</sup> The field of human-computer interface design finally had substantial guidelines for and definitions of usability and interface design methodology.

---

<sup>14</sup> John Canny, "The Future of Human-Computer Interaction." 26.

<sup>15</sup> Ibid., 26

<sup>16</sup> Ibid.

<sup>17</sup> J. Karat and C. M. Karat, "The Evolution of User-Centered Focus in the Human Computer Interaction Field," 532.

<sup>18</sup> Ibid., 534

<sup>19</sup> ANSI/HFES 200 Standard, Ergonomic Requirements for Software User Interfaces.

## II. Design Principles

In order to design usable interfaces, it is necessary to understand the process by which users perform actions. We define an action as a collection of simple behaviors performed in sequence to complete a very small portion of the larger task a software application facilitates. For example, the actions a user performs while utilizing a word processor include font selection, tab setting, and printing, but do not include the specific steps necessary to complete these actions, such as mouse clicks and key presses, nor do they include the broader activities these actions serve, such as writing a novel or research paper.

Actions may be divided into seven stages: forming the goal, forming the intention, specifying an action, executing the action, perceiving the state of the world, interpreting the state of the world, and evaluating the outcome.<sup>20</sup> When forming the *goal*, a user might decide to italicize a phrase of text. The *intention* might be to enter the text, then select it and ask the word processor to italicize the selected text. An alternative intention would be to set the word processor to italicize any text entered, then to enter the text, and finally set the word processor to stop italicizing text. To *specify an action*, the user must determine the physical behaviors required to satisfy the intention, such as clicking and dragging the mouse cursor over text then clicking on an icon at the top of the computer screen. After attempting to execute the specified action, the user gathers sensory data such as the configuration of light emitted by the computer screen, interprets that data to determine the application's state (in this example, to classify the format of the selected text), and finally determines whether or not the application's state satisfies the initial goal.

In order to form goals and intentions and specify actions, the user depends on a *conceptual model*, a mental representation of the application's available functions and of the actions the user must perform to utilize those functions. A good conceptual model enables the user to form viable goals and plan appropriate actions to obtain those goals. Good design provides users with good conceptual models.<sup>21</sup> A good conceptual model requires an accurate sense of the application's mapping of controls (command buttons, menu items, scroll bars, etc.) to functions. To provide this understanding, an application's controls should map to functions in

---

<sup>20</sup> Donald A. Norman, *The Design of Everyday Things* (New York: Doubleday, 1990), 48.

<sup>21</sup> *Ibid.*, 12-13

accordance with users' intuition,<sup>22</sup> there should be as close as possible to a one-to-one correspondence between controls and functions, and the application must offer clear, visible cues as to the functions of each of its controls.<sup>23</sup>

Appropriately chosen graphical images often make better visual cues than text because they take less time to interpret, form a stronger impression on memory, and contribute to users' motivation. All of an application's graphics should be stylistically consistent. Simpler graphics are easier to learn to recognize than complex graphics, as are graphics which metaphorically represent the functions of the controls they mark compared against abstract logos. Users will have difficulty interpreting graphics that employ culturally unfamiliar metaphors, and any graphics that require captions to understand offer no advantages over text alone and may consume excessive screen space.<sup>24</sup>

Providing one function per control facilitates graphical identification of controls, but it also makes controls easier to use and functions easier to remember. If a control has multiple functions, some of its functions may be difficult to discover, but a visible control with one function automatically reminds the user of its function simply by being visible. If a control's function depends on the application's state, the user may attempt to access one of the control's functions in the wrong context, and if a control's function may be changed by a sequence of user actions, the function becomes harder to learn to use and subject to greater risk of error. Multiple functions per control are likely to result in confusion and frustration.<sup>25</sup>

When multiple functions map to a single control, for example if a command button does different things if the user clicks on it while holding shift, or control, or alt, or some other combination of keys, the mapping of controls to functions becomes arbitrary.<sup>26</sup> Long term memory depends on deep understanding of information. The more meaningful information is, the more easily and effectively it is learned and later recalled.<sup>27</sup> Functions that map arbitrarily to

---

<sup>22</sup> Ibid., 23

<sup>23</sup> Ibid., 22

<sup>24</sup> Suzanne Watzman, "Visual Design Principles for Usable Interfaces" in *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications*, edited by Julie A. Jacko and Andrew Sears (New Jersey: Lawrence Erlbaum Associates, 2003), 273-274.

<sup>25</sup> D. Norman, *The Design of Everyday Things*, 22

<sup>26</sup> Ibid.

<sup>27</sup> Robert W. Proctor and Kim-Phuong L. Vu, "Human Information Processing: An Overview for Human-Computer Interaction" in *The Human-Computer Interaction Handbook*, 44

controls take longer to learn and when their misuse results in error, it is more difficult to determine the cause of the error and correct it.<sup>28</sup>

Long term memory retention is also enhanced by association. New information is learned better when it can be integrated with preexisting knowledge;<sup>29</sup> thus, the user can more readily learn to use controls that function as the user might expect them to upon inspection. The software designer can achieve this by mapping controls in accordance with physical analogies and standardized practices. Sliding bars, for example, should increase values as the user slides them upward or to the right.<sup>30</sup>

Given an adequate conceptual map, the user can successfully form goals and intentions and specify actions; however, the more complex the execution of a given action, the more opportunities for error that arise at every stage of action up to and including execution. Therefore, good design automates as much of the action as possible without taking necessary control away from the user. The user should have to interact with a minimal number of controls, and the operation of those controls should be simplified.<sup>31</sup> The quantity of controls, however, cannot always be reduced. Applications that require a large number of controls are easier to learn and use if controls are grouped into sets by similarity of function, and sets are visually and spatially separated on the screen.<sup>32</sup>

Once the user has executed an action, the application must provide adequate feedback for the user to determine what has happened as a result. The user requires this information to develop a good conceptual map of the application during the learning process, and without this information the experienced user cannot fix any errors resulting from misuse of the software. Further, insufficient information about the application's current state hinders the user's ability to appreciate the full range of available actions at any time. Visual cues are an effective means toward providing feedback<sup>33</sup> and should be utilized according to the same principles of clarity and organization that guide their usage as control markers. Auditory feedback may be useful in certain applications, but its use should be minimized since many users find it annoying, it

---

<sup>28</sup> D. Norman, *The Design of Everyday Things*, 67

<sup>29</sup> *Ibid.*, 115-116

<sup>30</sup> *Ibid.*, 23

<sup>31</sup> *Ibid.*, 51

<sup>32</sup> *Ibid.*, 94

<sup>33</sup> *Ibid.*, 99-100

demands much of the user's attention, and may be considered intrusive by those in close proximity to the user.<sup>34</sup>

Applications without sufficient, clear, and accurate feedback are unusable. They are also unexplorable. Users need applications to be explorable in order to form complete conceptual maps. Visible controls inspire curiosity about their functions. Salient, comprehensible feedback facilitates experimentation with controls by demonstrating their functionality. For an application to be explorable, however, it is also necessary to minimize the potential costs associated with errors. Users should be able to undo actions whenever possible. When it will not be possible to undo an action, the user must first receive a clear, thorough warning about the action's consequences and have the opportunity to cancel the action.<sup>35</sup>

A usable application, therefore, provides users with an accurate conceptual map through the use of clear, complete, immediate feedback and intuitive, one-to-one mapping of well-organized, visibly-identified controls to functions; however, these necessary principles are insufficient given that a diversity of users implies a diversity of interactive expectations and capabilities. To design software for an international market requires comprehensive research into every target culture,<sup>36</sup> but across all cultures it is vital to recognize limitations on usability imposed by disabilities. In software, the most common impairments leading to disabilities that interfere with usability may be classified as cognitive impairments, physical impairments, and perceptual impairments.

Cognitive impairments may be generalized in their effects, as with the 25% of the population whose intelligence is measured with an IQ at or below 90,<sup>37</sup> or specific in effect, as in the case of dementia, which affects the elderly and results in short-term memory loss.<sup>38</sup> An individual's cognitive abilities may change significantly from one moment to the next as well as over weeks.<sup>39</sup> Regardless of the nature of the specific impairment, users with cognitive impairments typically take more time to perform tasks than do users without cognitive impairments; thus, whenever possible, applications should allow users to perform tasks at their

---

<sup>34</sup> Ibid., 103

<sup>35</sup> Ibid., 183 - 184

<sup>36</sup> Aaron Marcus, "Global and Intercultural User-Interface Design" in *The Human-Computer Interaction Handbook*, 445

<sup>37</sup> Alan F Newell et al., "Information Technology for Cognitive Support" in *The Human-Computer Interaction Handbook*, 466.

<sup>38</sup> Ibid., 472

<sup>39</sup> Ibid., 465

own pace. Users' speed can be improved, of course, if information is presented in a simple, well-organized manner. Users with cognitive impairments also generally benefit when interfaces avoid complexity and suffer when asked to divide their attention or to focus their selective attention when presented with competing stimuli. Fortunately, consideration for the needs of users with cognitive impairments improves usability for all users.<sup>40</sup>

Consideration for the special needs of users with physical and perceptual impairments can also improve general usability, but these needs are different than those for users with cognitive impairments. Physical and perceptual impairments limit the range of hardware users may utilize to interact with an application. They may be the temporary results of the context in which a user is attempting to use the software,<sup>41</sup> such as involuntary hand movement in a speeding car on a gravel road,<sup>42</sup> or the results of a health condition which may be permanent or temporary and continuous or intermittent, and may change in severity for better or worse over time.<sup>43</sup> The most common physical impairments relevant to HCI affect the upper body and include: missing or misshapen limbs; impairments affecting bone and joint mobility as a result of arthritis, Parkinson's disease, or repetitive stress injuries; muscle weakness and/or paralysis as a result of amyotrophic lateral sclerosis (ALS), multiple sclerosis (MS), cerebral palsy, muscular dystrophy, or spinal cord injuries; and involuntary movements or difficulty controlling voluntary movements as a result of ALS, MS, brain injury, cerebral palsy, Parkinson's disease, or a stroke. Health conditions may lead to additional cognitive or perceptual impairments.<sup>44</sup> Perceptual impairments typically include impaired speech perception and visual and hearing impairments ranging from complete blindness and deafness to color-blindness and reduced hearing.<sup>45</sup>

A variety of innovative hardware and software solutions called assistive technologies are available to support users with physical and perceptual impairments. Speech synthesis programs such as screen readers are applications that translate text into audio output<sup>46</sup> and tactile displays

---

<sup>40</sup> Ibid., 467 – 468

<sup>41</sup> Andrew Sears and Mark Young, "Physical Disabilities and Computing Technologies: An Analysis of Impairments", in *The Human-Computer Interaction Handbook*, 488

<sup>42</sup> Ibid.

<sup>43</sup> Ibid., 484

<sup>44</sup> Ibid., 484 – 489

<sup>45</sup> Julie A. Jacko et al., "Perceptual Impairments and Computing Technologies" in *The Human-Computer Interaction Handbook*, 506 – 514

<sup>46</sup> Clare-Marie Karat et al. , "Conversational Interface Technologies" in *The Human-Computer Interaction Handbook*, 177

can alter their own shape to produce output in Braille.<sup>47</sup> Input devices can control cursor movement by tracking head or eye movement and speech recognition software can facilitate text entry.<sup>48</sup> Ongoing research in input devices that use electrophysiological data such as the electroencephalograph, which traces brain activity, the electromyograph, which measures a muscle's state, and galvanic skin resistance show particular promise for users with disabilities too severe to enable the use of existing devices.<sup>49</sup> Software engineers can assist users with physical and perceptual impairments by providing support for multimedia output and multimodal input.<sup>50</sup>

Unusable applications are often the product of the software engineer's failure to understand users with and without impairments. Engineers commonly expect users to understand an application as well as they themselves do; however, engineers form a complete conceptual map of the product before it is even developed but are often less expert than intended users in the tasks the application will help users perform.<sup>51</sup> The engineer must completely understand how intended users think about and perform tasks without software to identify the range of goals users will want to achieve using the new software and anticipate the intentions users will specify to accomplish those goals.<sup>52</sup> The engineer cannot produce usable software without a thorough understanding of the needs and abilities of intended users, the other resources that will be available to users, the effects of the environment in which users are likely to use the software,<sup>53</sup> and the improvements in task performance that the application will provide.<sup>54</sup> In order to gain this understanding, engineers must observe and interview intended users before designing the software, and continually test its usability throughout the development process.<sup>55</sup>

---

<sup>47</sup> Hiroo Iwata, "Haptic Interfaces" in *The Human-Computer Interaction Handbook*, 208 – 210

<sup>48</sup> Sears and Young, "Physical Disabilities and Computing Technologies", 498

<sup>49</sup> *Ibid.*, 495

<sup>50</sup> Jacko et al., "Perceptual Impairments and Computing Technologies", 515 – 516

<sup>51</sup> D. Norman, *The Design of Everyday Things*, 155 – 156

<sup>52</sup> Janice Redish and Dennis Wixon, "Task Analysis" in *The Human-Computer Interaction Handbook*, 925 – 926

<sup>53</sup> Traub, "Optimising human factors integration in system design", 96

<sup>54</sup> Redish and Wexon, "Task Analysis", 923

<sup>55</sup> *Ibid.*, 923 - 924

### III. Testing usability factors

How do you test for usability factors? There are many testing methodologies to choose from. The first task is to decide what information is needed from the testing. For example, is there a specific problem or element that needs testing or would an overall rating of user satisfaction be more useful? What is the budget for product testing? What is the nature of the interface being tested? Is it for educational purposes, the general public, or specialists? The answers to these and other questions help to determine which testing methods are appropriate for the project. Below we will present testing methods along with their requirements and advantages.

The simplest testing method for usability factors is user observation. In this testing scenario an observer watches a user using the software and collects information about the interaction. Kathleen Gomoll at Apple Computers writes, "Until I started observing users, I didn't know the excitement, the value, and the ease of involving users in design."<sup>56</sup> Gomoll has outlined a method for observing users in ten steps:<sup>57</sup>

1. Set up the observation...
2. Describe the purpose of the observation (in general terms)...
3. Tell the user that it's okay to quit at any time...
4. Talk about and demonstrate equipment in the room...
5. Explain how to 'think-aloud'...
6. Explain that you will not provide help...
7. Describe tasks and introduce the product...
8. Ask if there are any questions before you start; begin the observation...
9. Conclude the observation...
10. Use the results...

It is easy to imagine what most of these steps entail, but one that may not be clear to the reader is the 'think-aloud' technique. In this testing scenario the user is not only using the software and executing the outlined tasks, but also verbalizing every thought and question she has and every action she is taking in the process.<sup>58</sup> These comments are recorded by the observer to help give a deeper understanding of where any usability problems began and where

---

<sup>56</sup> Kathleen Gomoll, "Some Techniques for Observing Users." *The Art of Human-Computer Interface Design*. Brenda Laurel, Ed. (Massachusetts: Addison-Wesley Publishing Company, 1990) 85.

<sup>57</sup> Ibid.

<sup>58</sup> Kent L. Norman and Emanuele Panizzi. "Levels of Automation and User Participation in Usability Testing." *Interacting with Computers* 18. (2006): 248.

improvements can be made. The observer does not even have to be in the room during such an observation; user and observer can connect via teleconferencing technologies.<sup>59</sup>

Like all testing methods, 'think-aloud' is not perfect. Researchers have found that requiring the users to explain what they are doing while performing a task changes their experience with the software, which means data about users in the lab will not match users' experience with the software in everyday life. But without using the think-aloud technique, it is hard to gather information about the user's unconscious decisions and thoughts through traditional observations and automated testing methods. As Norman writes: "A substantial part of the user's interaction with a system is looking at or listening to the interface without generating any feedback to the system. It is important to know where the focus of attention is for the user because it involves the locus of information acquisition or the point of decision-making."<sup>60</sup> As a solution to the problems of the think-aloud method, a new technology has been adopted by usability researchers to gather some of the same information: eye-tracking.

Researchers Crowe and Narayanan assert that eye tracking "can provide detailed data on the allocation and shifts of users' visual attention across interface entities."<sup>61</sup> Unfortunately, eye-tracking systems are expensive, probably too expensive for any but large software companies or usability laboratories.<sup>62</sup> In addition to the prohibitively high cost for hardware and software, eye-tracking devices require that the usability specialists performing the tests have some training in analyzing eye-tracking data and facility with the hardware. The fact that eye-tracking provides useful information but is not an option for many designers has led Norman et al. to design a simple alternative: the mouse tracker.

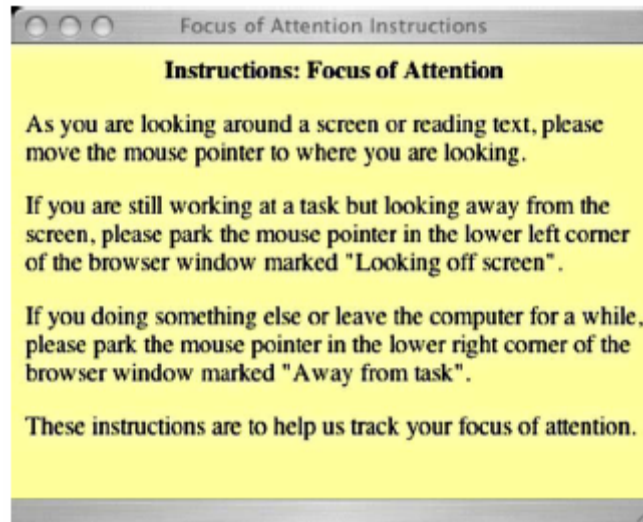
---

<sup>59</sup> Ibid. 253

<sup>60</sup> Ibid., 251-252

<sup>61</sup> Eric C. Crowe and N Hari Narayanan. "Comparing Interface Based on What Users Watch and Do." *Eye Tracking Research and Applications Symposium*. (2000): 29.

<sup>62</sup> As an example, the price of one eye-tracking system from Arrington Research is about \$12,000 for a complete hardware and software package. This is for one user. This was among the least expensive systems found during a non-exhaustive internet search for eye-tracking systems for research. Source: <http://www.arringtonresearch.com/prices.html>. (Accessed April 7, 2007).



**Figure 1: An Alternative to Eye-tracking proposed by Kent L. Norman et al.<sup>63</sup>**

According to Norman, "Mouse tracking is easy to record and will serve as a proxy for the user's focus of attention rather than eye tracking."<sup>64</sup> This tool is still in development, but promises to be a great alternative while eye-tracking systems remain unaffordable.

Usability specialist Jakob Nielsen maintains that "usability tests do not have to be complex to be effective."<sup>65</sup> The ability to rapidly produce a prototype is important for efficient user-centered design, especially for small companies without large development teams. Without complex algorithms, surveys or expensive mockups, Nielsen and his collaborators used index cards and paper versions of their interface for user testing, and have found these methods to be as effective as other more expensive methods for assessing usability and improving the design. It should be noted that, as Nielsen writes, these methods "rely more on the interface engineer's ability to observe users and interpret results."<sup>66</sup> Nielsen is a veteran usability researcher, and is therefore capable of understanding usability problems through informal observation. This is not to say that designers without knowledge of usability design principles can make "common sense" evaluations of software in the same manner. The use of informal methods and quick, inexpensive models is only useful for usability testing if the observer knows what to look for. Since user

---

<sup>63</sup> Norman and Panizzi. "Levels of Automation and User Participation in Usability Testing." 262.

<sup>64</sup> *Ibid.*, 263

<sup>65</sup> Jakob Nielsen and Bill Curtis. "Applying discount Usability Engineering." *IEEE Software* 12. 1 (Jan. 1995): 98.

<sup>66</sup> *Ibid.*

observation can be costly and time-consuming, usability specialists have developed alternatives, such as automated testing.

If all that is desired from the usability test is a general overview of user satisfaction, or knowledge of whether glaring usability problems exist in the program, then a user survey will suffice. Two examples of such general user satisfaction tests are the Questionnaire for User Interaction Satisfaction (QUIS)<sup>67</sup> and the Software Usability Measurement Inventory (SUMI)<sup>68</sup>, which have been proven to give accurate usability and satisfaction ratings. These surveys offer mostly subjective evaluations of the program and cannot identify the cause of the problems. If quantitative and more problem-specific data is needed, then an alternative method is to keep a "history file" of the user's interaction with the system, like button clicks and typed text. This can be helpful and provide a lot of data without an observer having to be present, but there are drawbacks to this method. Norman observes that significant programming may be required to detect navigation patterns "...such as failure to efficiently navigate through a site rather than use the back button."<sup>69</sup> Additionally, if none of the user's thoughts or questions are recorded along with this data, it can be hard to know what he was trying to accomplish at any given time, and therefore to know whether he experienced difficulty in the process<sup>70</sup> which can make it hard to evaluate the effectiveness of the interface design. A more elaborate means of automated testing is to use a program like the Noldus Observer<sup>71</sup>, which can "code behavior events, record times, and associate events with video capture."<sup>72</sup> This will result in more useful data than a mere history file; however, as Norman observes, the program does not totally automate user testing. "This and other programs... still require more manual labor than seems appropriate."<sup>73</sup> Technology for user observation has not become fully automated, and if automation is even possible, there is much research that needs to be done.

Automated testing can also be quite costly, as with the use of artificial intelligence to simulate users. According to Norman, the goal is to replace human test subjects with "surrogate

---

<sup>67</sup> John P.Chin et al.. "Development of a Tool Measuring User Satisfaction of the Human-Computer Interface." *Proceedings of the SIGCHI conference on Human factors in computing systems*. (1988): 213-218.

<sup>68</sup> J. Kirakowski and M. Corbett. "SUMI: the Software Usability Measurement Inventory." *British Journal of Educational Technology* 24.3 (Sept. 1993): 210-212.

<sup>69</sup> Norman and Panizzi, "Levels of Automation and User Participation in Usability Testing," 251.

<sup>70</sup> Ibid.

<sup>71</sup> L. L. Noldus, "Software Tools for Collection and Analysis of Observational Data." *Proceedings of the Eighth International Conference on Human-Computer Interaction 2* (1999): 1114-1118.

<sup>72</sup> Norman and Panizzi. "Levels of Automation and User Participation in Usability Testing." 252.

<sup>73</sup> Ibid.

users in the form of artificial intelligent user agents that model the perceptual and cognitive processes of real users."<sup>74</sup> Although the use of AI to replace humans in usability testing elicits both enthusiasm and fear, Norman states frankly that these projects are "doomed to fail for a variety of reasons... Past and current attempts have been extremely costly, overly simplistic, and totally specific to the task and application,"<sup>75</sup> which means one should not expect to see widespread use of such automated testing in the near future.

The usability testing techniques we have described up to this point have been mostly focused on software for a professional environment, used primarily by adults. There is a lot of software that doesn't fall into this category, such as educational software for children. In addition to the usual criteria for designing an interface, software for children must be "intuitive and not distract the user from achieving their objectives,"<sup>76</sup> and fun! One can imagine that children may not be able to sit through traditional usability tests, such as observations, and may not be able to analyze their experience using a program in such a way that would benefit software designers. Gavin Sim et. al at the University of Lancashire in the UK studied the importance of fun in educational software for children and developed two useful testing techniques: a "Smileyometer" for subjective rating and a "Fun Sorter" to rank different applications according to fun, usability, and educational value.<sup>77</sup>



**Figure 2: the Smileyometer<sup>78</sup>**

Instead of choosing a number from 1 to 5, as with subjective rating tests for adults, the children chose a smiley face that best represented their feeling toward each application before and after use. The "Fun Sorter", on the other hand, was used to compare different programs; the children

---

<sup>74</sup> Ibid.

<sup>75</sup> Ibid.

<sup>76</sup> Gavin Sim et al.. "All work and no play: Measuring fun, usability, and learning in software for children." *Computers & Education* 26. (2006): 237.

<sup>77</sup> Ibid., 241-242

<sup>78</sup> Ibid., 241

pasted screen-shots of each program on a sheet of paper in order according to agreement with the statements provided, for example, "I think my teacher would choose this software" or "easiest to use.... hardest to use."<sup>79</sup> What Sim et al. found was that the children, at 7 to 8 years old, were able to give significant feedback on fun, usability and educational value using these methods; the authors further suggest that the "Fun Sorter" might have value for testing software characteristics beyond just fun, and could be used for testing all kinds of software for children.<sup>80</sup>

It is important to note that although these testing methods have been outlined individually they are most often used together; in fact, combining the data collected from different kinds of tests is the best way to get a complete picture of users' experiences, including their conscious and unconscious reactions, and collect subjective as well as objective data.

#### IV. Conclusion: The Future of Human-computer Interface Design and Usability

As mentioned above, no major innovations in human-computer interface design have been made since the development of the WIMP interface back in 1976. John Canny at the University of California, Berkeley, predicts that this is about to change.<sup>81</sup> Consider computing devices common to most households: cell phones, digital cable boxes for the TV, game consoles, and even interactive security and control systems. All of these devices are using the same kind of interface common to the first WIMP computer, the Star. The intended use of the WIMP interface was office work like word processing and accessing files, but today's devices are required to perform many diverse tasks and require many more complex and rapid interactions with the user. What innovations in human-computer interaction are currently available on the market that could potentially replace the WIMP model in our everyday devices? Three examples are multi-touch screens, eye-trackers, and multimodal input devices.

Jeff Han at the Media Research Laboratory at New York University has developed a multi-touch screen that allows users to interact with virtual windows and objects in a method similar to touching real objects. As the cost of eye-tracking devices decreases, this technology could prove useful not only for users with disabilities and for usability testing, but also for everyday users as a replacement for current pointing and navigation devices. The controller to

---

<sup>79</sup> Ibid., 242.

<sup>80</sup> Ibid., 246

<sup>81</sup> John Canny. "The Future of Human-Computer Interaction." *ACM Queue* 4. 6 (Jul.-Aug. 2006): 27.

the Nintendo Wii is an example of a multi-modal input device, as it captures both button presses and motion. The Wii's success with all kinds of users<sup>82</sup> may lead to the development of similar devices for everyday use. These three devices comprise an incomplete sample of current innovations in human-computer interface design.

The most important point for software developers to take from the vast body of research on usability and human factors is that a thorough understanding of the user is the foundation for good interface design. Testing an interface with real users throughout the design process is the best way to build usable software and avoid the problems and costs of poorly designed interfaces. As we have shown, clear guidelines exist for such details as buttons and menu design, and many testing methods are available for large and small budgets. While a simple overview of these ideas is not enough to ensure a software designer will avoid all possible usability problems, knowledge of the importance of usability factors in human-computer interface design will lead to wider acknowledgement of the importance of usability and a reduction in the number of poorly designed interfaces to which we will all be subjected in our everyday lives.

---

<sup>82</sup> Matt Richtel et al.. "Games That Sell While Others Languish; Nintendo's Wii, Radiating Fun, Is Eclipsing Sony." *The New York Times*, Jan 31, 2007, national edition, <http://nytimes.com> (Accessed April 13, 2007).

References:

- Aspin, Les. "Witness to Iran Flight 655." *New York Times*, 18 November 1988, national edition, <<http://www.nytimes.com>> [cited 15 April 2007].
- Canny, John. "The Future of Human-Computer Interaction." *ACM Queue* 4. 6 (Jul.-Aug. 2006): 25-32.
- Chin, John P., Virginia A. Diehl and Kent L. Norman. "Development of a Tool Measuring User Satisfaction of the Human-Computer Interface." *Proceedings of the SIGCHI conference on Human factors in computing systems*. (1988): 213-218.
- Filman, Robert E. "Interface Pains." *IEEE Internet Computing* 8. 5 (Sept.-Oct. 2004): 4-6.
- Gomoll, Kathleen. "Some Techniques for Observing Users." *The Art of Human-Computer Interface Design*. Brenda Laurel, Ed. Massachusetts: Addison-Wesley Publishing Company, 1990. 85-90.
- Han, Jeff Y. "Low-Cost Multi-Touch Sensing Through Frustrated Total Internal Reflection." *Proceedings of the 18<sup>th</sup> Annual ACM Symposium on User Interface Software and Technology*. (Oct. 2005): 115-118.
- Iivari, Netta. "Usability specialists - 'A Mommy Mob', 'Realistic Humanists' or 'Staid Researchers'? An analysis of usability work in the software product development." *Interact*. (2005): 418-430.
- Jacko, Julie, and Andrew Sears, Eds. *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications*. New Jersey: Lawrence Erlbaum Associates, 2003.
- Karat, J. C. M. Karat. "The Evolution of User-Centered Focus in the Human Computer Interaction Field." *IBM Systems Journal* 42. 4 (2003): 532-540.
- Kirakowski, J., and M. Corbett. "SUMI: the Software Usability Measurement Inventory." *British Journal of Educational Technology* 24. 3 (Sept. 1993): 210-212.
- Nielson, Christian M., Michael Overgaard, Michael B Pedersen, and Jan Stage. "Feedback from usability evaluation to user interface design: Are usability reports any good?" *Interact*. (2005) 391-404.
- Nielsen, Jakob, and Bill Curtis. "Applying discount Usability Engineering." *IEEE Software* 12. 1 (Jan. 1995): 98-100.
- Norman, Donald A. *The Design of Everyday Things*. New York: Doubleday. 1990.

- Norman, Kent L., and Emanuele Panizzi. "Levels of Automation and User Participation in Usability Testing." *Interacting with Computers* 18. (2006): 246-264.
- Reingold, Howard. "An Interview with Don Norman." *The Art of Human-Computer Interface Design*. Brenda Laurel, Ed. Massachusetts: Addison-Wesley Publishing Company, 1990. 5-10.
- Richtel, Matt et al. "Games That Sell While Others Languish; Nintendo's Wii, Radiating Fun, Is Eclipsing Sony." *New York Times*, 31 January 2007, national edition, <<http://nytimes.com>> [cited 13 April 2007].
- Sim, Gavin, Stuart MacFarlane, and Janet Read. "All work and no play: Measuring fun, usability, and learning in software for children." *Computers & Education* 26. (2006): 235-248.
- Stewart, Tom. "Ergonomics standards concerning human-system interaction: Visual displays, controls and environmental requirements." *Applied Ergonomics* 26. 4 (1995): 271-274.
- Sutcliffe, Alistair, and Antonella De Angeli. "Assessing Interaction Styles in Web User Interfaces." *INTERACT*. (2005): 405-417.
- Traub, Paul. "Optimising human factors integration in system design." *Engineering Management Journal* 6. 2 (Apr. 1996): 92-98.
- Vertelney, Laurie, and Sue Booker. "Designing the Whole-Product User Interface." *The Art of Human-Computer Interface Design*. Brenda Laurel, Ed. Massachusetts: Addison-Wesley Publishing Company, 1990. 57-63.

### Appendix: Task Summary

Audrey Troutt wrote the Abstract and sections I, III, and IV.

Daniel Sheiner wrote section II.

Both Audrey and Daniel revised the entire document.