

**Review Paper:**  
**Human Computer Interaction Design**  
*Implications for Education*

Jeffrey Anderson  
ASU EDT 591 - Atkinson  
“Current Trends in Instructional Technology”  
Fall 2007

## Introduction

Human Computer Interaction (HCI) tends to have a broad range of topics associated with it, which leads it to have a number of definitions. One that is broadly agreed upon is "...a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them." (Brown & Abbie, 2004) HCI has sometimes in the past been synonymous with Computer Human Interaction (CHI), but with the turn of the millennium, human-centered computing has been a prime focus with researchers, therefore the more typical reference is HCI which puts the "human" before the "computer" as the emphasis.

HCI as a field of study has its roots in the workplace. Computers have long held the role of being an aid to the work of mankind, and what of the better methods to make a human's job more efficient than to have a computer helping them complete their work. A computer is computationally more efficient than a person. However, the person using the machine isn't always adept at figuring out how to tell the computer sitting in front of them *how* to assist them in their work. The earliest computers that showed up in the mainstream work environment as desktop machines were monopolized by industry giants such as IBM and Apple. Therefore to maintain their edge, these companies invested time and money to make their respective hardware and software easier for people to use; especially for those individuals with minimal to no experience with computing since it was relatively new at the time. Only until computers had been introduced into people's homes and schools have they had the impact on society that they do today; more especially if the computer in question has Internet access. As a result, HCI is now a factor that affects the economy of the world as it pertains to areas of personal finance, business, entertainment, and of course education.

Closely related to HCI is the topic of Industrial Design. Not to be confused with *Instructional Design* (ID), Industrial Design is a topic that refers to how humans interact with tangible products. The reason Industrial Design and HCI are closely related is that often a computing device has to be contained within an enclosure that is easy for a human to approach and have productive interaction. Consequently, a measuring of how effective a computer interface is to a human being is *usability*.

Usability has no defined metric in studies involving HCI and Industrial design simply for the fact that there are so many varieties of computing products, interfaces, and scopes of usage for them. Therefore for the purposes of this review, a pseudo definition will be introduced for the sake of discussion: A computer interface is considered usable if said interface is intuitive enough for a human to approach and use with minimal intervention from other authoritative sources (i.e. help systems, expert users, manuals, etc.). This does not mean that authoritative sources aren't desirable in the learning process, rather it simply means that once all prior knowledge is obtained and the user is instructed properly on how to use a computer interface, that these sources are secondary and the less they are depended upon the more usable that interface is for that particular person or intended audience for that matter.

The focus of this review will address two major problems by asking the following questions: **“How is the topic of HCI related to educational technology today?”**, and **“How do principles of HCI and Industrial design carry over to instruction.”** The review will begin with a brief history of computers and their respective interfaces, and a look into the future concerning the same. Using this information as a background, the review will provide suggestions and draw conclusions on how to address these stated issues.

## History of Computer Interfaces

Computers had been around long before they started showing up on people's desks in the workplace and at home. One of the very first well-known computers from history (1946 to be exact) was ENIAC (Electronic Numerical Integrator and Computer). One of its primary purposes was to compute artillery trajectories for the military. ENIAC was so large that it took up entire rooms to house it. People interacted with the machine only in the way of programming it, and recording the output or results of the programs. They did this by inserting cards with punched holes, connecting electronic circuits with cable connections, flipping switches and subsequently collecting the output of computations on punched cards (Muuss, n.d.).

Mainframe computers were the next iteration of machines that took up large spaces. These eventually saw their way into businesses. Like ENIAC, the operators of these machines often found themselves feeding the machines programs by means of cards with punched holes or encoded on magnetic tape, and then recording the results of running those programs on paper printouts. Eventually the user interface to these mainframe computers transitioned into a single computer terminal with a keyboard and monitor not unlike seen on modern personal computers. Also the machines were only affordable to businesses with large sums of cash.

The first computer meant to be affordable to an average person was the ALTAIR computer. The machine itself wasn't very impressive to look at. It was a simple blue box with a series of switches. It didn't come with an input or output devices (keyboard or monitor) that the normal every day person could use. The ALTAIR computer's target audience was that of computing enthusiasts and hobbyists who reveled in the idea that they could have a real computer sitting on their table in their own home. These users would spend their time on the machine by flipping the switches in succession in order to program it to do mundane things like

simple mathematics problems, and occasionally get it to play simple musical riffs via its one-note beeping internal speaker. Only until a small, then-unheard-of company called “Microsoft” offered to sell its BASIC programming language to the ALTAIR’s manufacturer did it receive any further notoriety.

Early mainframe computers used in the military and businesses eventually employed a keyboard and monitor input and output device. The interface to those types of machine was known as **Command Line Interface** (CLI), because to get the machine to do anything its operator would type instructions to the machine on the keyboard in the form of words or commands. By the time home computers came on the market, this was virtually the only method of interacting with a computer. In order to become adept at working with such a device, the user had to be familiar with what commands the computer could understand, and then type and enter those commands into the machine using a keyboard without making mistakes. By this time, software programs could be stored on external media such as ROM cartridges, cassette tapes and floppy disk drives. No longer was a user restricted to what the computer’s base operating system was designed to do. It could load programs written and shared by other programmers. Now not only did computers exist in people’s homes, they also started showing up in school classrooms. As a result, children were interacting with computers. People who grew up during the time that ENIAC was first put to use during World War II could now own and place a computer in their living room that was many times more powerful than ENIAC, and their children were the beneficiaries. Popular brand name home computers of the age were Atari, Commodore, Colecovision, Tandy, IBM, and of course Apple.

Apple computers were revolutionary in their day due to their relative affordability (under \$1300 US for the base model) and their appliance-like appeal which also made them the de-facto

standard computer in American education (“Apple II Series”, 2007). Apple computers were also the first widely used machines to make use of a graphical user interface (GUI). In contrast to the command line interface (CLI), computer users could use a mouse as an input device and interact with bitmapped icons on a computer monitor that were metaphors for computer-related tasks. An example of this would be a graphical virtual desktop with documents represented as paper icons strewn across it; along with a trash can icon which would allow a user to destroy an unwanted program or document by dragging its respective icon over the trash can icon (throwing away the trash). This eventually became Apple’s best selling computer line, the Macintosh platform. This type of interaction in HCI has also been coined as the WIMP (Windows, Icons, Mouse, Pointer) interface (Canny, 2006).

Following suit for the IBM PC based machines was Microsoft who historically sold and signed a deal to write the operating system for IBM’s new PC computers when it in fact had none. Its parallel to Apple’s early CLI, Apple DOS, was Microsoft DOS or MS-DOS. At this early stage, Microsoft as a company was in the software writing business. In addition to its own MS-DOS platform for IBM, Microsoft wrote software for other companies as well. Apple employed Microsoft to write software for the early Macintosh platform. Not long after, Microsoft upgraded its MS-DOS line to include a GUI system called Windows which also contained bitmapped icons and represented running programs as windows and documents on a virtual desktop. Notwithstanding the legal implications of Microsoft borrowing the GUI concept for its own line of computers, this single act spurned a competition between Apple and Microsoft that still exists presently. While the Macintosh platform has been a success in computing to this day, eventually the computer market gave way to Microsoft’s Windows platform being the predominantly pervasive computing environment in homes and schools. The GUI is the way that

most people new to computing begin their experience with interfacing with computers, and they do it predominantly on Microsoft's Windows platform. Therefore the GUI is still the central focus to HCI design as it pertains to home use, education, training, and workplace productivity.

As computer usage with the GUI as its main interface for users was monumental, the GUI also changed the way that engineers wrote software for the various computing platforms. Under a CLI, a software program was comprised of computer code stored in text files that are fed to a second program called a compiler. The compiler eventually creates a working executable program file. Under a GUI development system, the process is a lot more visual. Now software engineers have an array of graphical tools used to visually draw the look and feel of a program out on the screen using a mouse, and a special program called a visual integrated development environment (IDE). A modern IDE of this type is responsible for taking these visual drawing cues and generating the code needed to create the actual finished software on behalf of the software designer. After the interface to a program is designed visually, the engineer can focus on the logic behind the interface by looking at and enhancing the code that is generated by the IDE and add new and more focused features to the program. This has huge implications to HCI in the fact that separation of tasks can now include a dedicated HCI expert or team of experts to design the interface, and dedicated software engineers to focus on performance and logic of the software design. The nature of the IDE is such that common interface elements appear as icons on screen and can be drawn and arranged according to principles of screen design. One of the earliest IDEs to provide a visual programming experience was Microsoft's Visual Basic. This was and still is a programmer's productivity tool, and allows developers of varying backgrounds, domains, and skill levels to create software quickly for Windows computers that took on the look and feel of the Windows environment itself ("Visual Basic", 2007). Since it is a relatively easy

language to learn, some of the earliest educational software programs for windows computers were done using Visual Basic. An equivalent IDE on modern Macintosh operating systems is called XCode. However software development on a Macintosh computer using XCode is considered more elite among programmers because the languages it supports are syntactically less verbose. Much early educational software on the Macintosh platform was done with a now defunct system called HyperCard, which historically is one of the first hypermedia systems on a computer (“HyperCard”, 2007).

Much in the way of educational content delivered via computer interfaces has to do with the Internet, and the web browser software with which a learner chooses to access it. The appeal of the Internet is that software designed to access it can display unlimited numbers of remote web sites written in HTML (Hypertext Markup Language), and transform them into a user interface to be interacted with. Much like other visual programming languages, HTML code describes a document structure that includes common user interface elements such as formatted text layouts, text input boxes, buttons, checkboxes, etc. Thus much of the same interfaces we see on common desktop applications written in other languages can be had inside the web browser, with lots of flexibility. This has an even greater bearing on HCI design for several reasons. The main reason is that since HTML code is relatively verbose, just about anyone can create a web site. Because not all HTML web developers are versed in good principles of screen design, there are many web interfaces that are not deemed usable by measures of good HCI design. Another reason is that users have a choice as to what browser software they wish to use to access HTML documents on the web. Often the everyday user will just use what is placed in front of them, and even more commonly won't be aware that they have a choice. The earliest web browser consisted of a simple plain text interface. The first graphical web browser capable of displaying

images and color was known as NCSA Mosaic, and is the precursor to all other modern web browsers today. Because of competition in the early days of mainstream Internet use, the browser was the key focus to platform dominance. NCSA Mosaic first gave birth to the popular Netscape browser. Microsoft historically lagged behind Netscape at first in terms of browser development. Since Netscape was also available to Macintosh computers, and a small variety of other computing platforms, Microsoft saw its platform dominance in the marketplace threatened. It spearheaded its own web browser also based on NCSA Mosaic, called Internet Explorer. The first versions weren't much to behold, but eventually Internet Explorer overtook Netscape as the dominant pervasive web browser that most people used, primarily because Microsoft began bundling Internet Explorer for free in every copy of Windows that it sold. Since Netscape charged their users money for the use of their browser, they had to start offering it for free in order to compete for platform ubiquity. Eventually, Netscape ran out of cash and subsequently decided to give away the source code for its browser to the open source community under the name "Mozilla" in order to keep the project alive and compete with Internet Explorer ("Mozilla", 2007). The open source movement in general allows anyone with the knowledge to do so to gain access to computer code for a piece of software and then compile the code and build the software for free. Modifications can be made to the original code with the stipulation that re-released versions of the software built with it must be distributed in turn with its source code. Netscape essentially became the Mozilla group. This spawned a number of free web browser products that to this day includes the popular Firefox browser ("Browser Wars", 2007).

What all of this means to HCI and its implications on education is that users have a choice as to what platform and accompanying software with which they will choose to access the Internet. It also impacts website developers (including those authoring educational content), as

they must be cognizant as to their target audience, and what browser they might choose to access their content. Different browser platforms render web sites differently according to what types of technology and multimedia plug-ins are available to the platform (Windows or Macintosh, etc.), what device is being used to access the site (desktop computer, mobile phone, PDA, etc.), and also how competent the web site designer is adept at structuring the HTML code behind the websites they design to be aware of this fact. Web usability dictates that a user will return to a web site/interface if they had a good productive experience by doing so, and therefore they should not be “punished” with a presented broken interface for simply visiting a website. If a site developer improperly structures their code, the browser could break the intended design, or worse, the browser could display an error message that the every day user may not know how to interpret or handle. This reduces the chance that user will visit or access that particular web site again.

Although the Internet has had a big hand in the development of HCI design principles it has not completely phased out traditional desktop computing and computer-based training. Several authoring environments such as Microsoft PowerPoint, Apple Keynote, and many Adobe Systems products (Director, Flash, Captivate) are yet entirely capable of providing rich learning experiences to the people that access products designed with those tools. Naturally, principles of good screen design apply to those environments as well.

## **The Future of Computer Interfaces**

As long as computers are providing simple 2-dimensional screen-based experiences to humans, there will always be a need for well designed screen-based e-learning products. A current trend today is to provide Internet users with web pages that employ a desktop-like response experience. The Internet has been traditionally a request, refresh, display experience in

the sense that a user visits a web site or clicks on a hyperlink to a new document, the browser must clear itself of the old content, refresh with new content, and display it to the user. This process is dependent on the user's network connection speed. If it is a slow connection, the response time to a user interaction can be detrimental to the overall experience. A current trend already taking place is to make the experience of interacting with a web site more like the experience of interacting with a traditional desktop-based piece of software. More recently employed technologies such as AJAX (Asynchronous JavaScript and XML) and DHTML (Dynamic HTML) allow for a web developer to design their web sites with functionality that changes/updates only the necessary parts of a web page when a link is clicked. This eliminates a web page's response time for change, since the whole screen doesn't have to reload.

Additionally, existing multimedia technologies now have the capability of filling the entire web page with a full screen design that behaves like a traditional desktop application, as opposed to a scrolling document filled with text and hyperlinks. This type of web experience is being referred to as a Rich Internet Application or RIA.

From a developer/practitioner standpoint future screen-based HCI trends involve the convergence of the desktop user experience, and the web page user experience. Standalone applications have the capability of being designed to access and aggregate data that would normally appear inside of a web page using a cross-platform runtime system. Cross-platform means that an application can be developed once and run on two or more competing platforms (i.e. Windows and Macintosh). Developers of these types of media incorporate end-to-end user experience, from screen design, access and display of data and content via the Internet, interaction, and appropriate response. Technologies from this space include Apple's iPhone and iPod Touch products with its touch screen interface ("Apple – iPod Touch – Webapps", 2007),

Mozilla's XULRunner platform ("XULRunner – MDC", 2007), and Adobe Systems AIR platform ("Adobe Air", 2007).

From a general futurist standpoint, HCI design has many areas in which it can progress. Extending beyond the WIMP interface (Canny, 2006), new ways of interacting with a computer is being focused on devices and software being contextually aware of their users. For instance, if someone is using their cell phone inside a restaurant, the cell phone could be designed to be contextually aware that it is in the restaurant, and thus call up the restaurant's menu system so that person can order what they would like to eat without having to speak to a server. A step beyond is allowing the person to speak in real time to the restaurant's ordering system. A virtual server takes the order, and the cell phone completes the transaction with a debit card payment stored in the owner's profile. This has implications in Educational Technology in the sense that every student's cell phone could in theory become a clicker response device and is aware of the class the student is taking and configures itself for the class when the student enters the classroom, with no intervention from the student except for the intended clicker usage.

Other areas of growth and new means of interacting with machines include the use of speech interfaces (which are progressively getting better each iteration), and haptic interfaces where the user can selectively position devices or hands and fingers in space so as to manipulate data on the machine. This is already being seen today in modern entertainment using Nintendo's Wii gaming system. It employs the use of special remote control devices that the system can detect in motion and position, resulting in the ability to update and manipulate onscreen characters and other game elements.

Finally, the aspect of computers and computing devices taking on human-like personalities much in the way that science fiction/fantasy media and Hollywood movies of the

same genre is a valid area of research, invariably could improve the response of humans interacting with the technology that a computer interface of this type could provide. Studies even indicate that humans have increasingly been treating computers as social entities since the last decade (Moon & Nass, 1996; Wenger, 1991).

## **Addressing the Issues**

Given the history and context for use of HCI in general, the following questions can now be addressed in the context of education:

1. How should HCI be approached in educational technology?
2. How do principles of HCI and industrial design carry over to instruction?

HCI is a key concept to educational technology as it is, since the computer has become a mainstay in the classroom and at home. Since a big portion of technology used in learning involves desktop computers, interfaces that promote learning through the use of proper screen design principles are paramount. Educational content developers must consider principles of human behavior and reaction to the mediums by which members of their intended audiences will access their content. For instance, an interface for a student in K-6 would likely need to be adjusted for someone enrolled in a remedial class on the same content. Also, it is important to consider age itself, not necessarily grade level alone when designing interfaces for the purposes of learning. There are implications when designing for children and for adults separately, and even age ranges within adults. Often it is younger designers creating interfaces for older users, and this must be considered in the design of interfaces to educational content as well as the content itself (Hawthorn, 2007).

Additionally, questions such as allowing an interface to be adjustable to accommodate different learners approach to consuming information, and transfer of knowledge is directly

attributable to principles of HCI. Discoveries and categorizations of human learning styles are now able to be identified. Individual mind styles can be taken into account when a designer of computer aided instruction is sitting down to organize modules of instruction. For instance learners who prefer concrete sequential (CS) instruction as opposed to that of a random abstract (RA) flavor might appreciate an interface that automatically adapts to their preferred method of cognitive information processing (Ross, J., & Schulz, R., 1999).

Given the progression of technology in general, the **study of HCI as a topic** is also now becoming prevalent for students of web design, industrial design, computer science, and educational technology. This is a broad range of topics that is multidisciplinary in nature. As an example, computer science curriculum is heavily engineering based, while web design has traditionally been based in arts curriculum with an emphasis in computing technologies. These two areas respectively constitute a logical quantitative discipline, and an artistic creative field. Yet, these two skill sets are beginning to blend and are also in relatively high demand with the pervasive use of the Internet in our society. Additionally, HCI design and industrial product design are becoming more related fields in education. Emerging evidence suggests that educational programs from multiple disciplines are starting to converge so that students entering those programs will be better prepared for the workplace of tomorrow (Brown & Sugar, 2004; Faiola, 2007; Kolko, 2004). Additionally only recently have we seen a “practice what is preached” methodology to teaching HCI design principles in the curriculum of HCI research itself (Parush, 2006).

As for how principles of HCI and Industrial design carry over to principles of Instructional Design (ID), if the mode of instruction is carried out via a screen device, this leads the instructional designer to the task of knowing the environment to which their learners will be

associated. Electronic Performance Support Systems are an example of a device that isn't necessarily a computer, but still designed and capable of delivering e-Learning content. ("Electronic Performance Support Systems", 2007). In this case, the design of the interface and subsequent user interaction has to be considered at both the **software and the hardware** levels in addition to other human factors present in a given situation. As a concrete example, perhaps a municipal power plant has frequent turnover at a position put in charge of servicing a generator with a proprietary on-board screen-based control console. To improve time spent on training, the machine's designers contracted out to a design firm to implement onboard on-the-job help in the console itself. One of the considerations given to the design team is that the types of people accessing the console would be wearing safety equipment which included heavy gloves. Therefore, the interface could not likely be touch-screen dependent, because that type of interface depends on human's fingertips being able to touch the screen with no obstruction. Also, in this hypothesized scenario, the type of screen being used should be able to be easily read in all lighting conditions. These types of considerations would be all encompassed in the role of the instructional designer contracted to complete the hardware and software interface for this project so that it would be maximally effective in achieving its learning outcome for its users.

## **Conclusions and Wrap-up**

In this review, the role that Human Computer Interaction plays in Education has been addressed. Trends in the development of computing technology over time have been taken into effect, as well as those in the present, and into the future. The challenges, problems and plausible solutions to those issues in how the subject of how educational technology is taught and the role that HCI plays in instructional design have been addressed. From this it is hopefully seen that

HCI is a dominant factor in the presentation and mediation of instruction in the traditional ID processes of instructional analysis, design, development, implementation, and evaluation.

## References

1. Apple – iPod Touch – Webapps (2007, Dec 4) Retrieved December 4, 2007 from <http://www.apple.com/ipodtouch/webapps/>
2. Apple II Series. (2007, Dec 3). In *Wikipedia, The free encyclopedia*. Retrieved December 3, 2007, from [http://en.wikipedia.org/wiki/Apple\\_II](http://en.wikipedia.org/wiki/Apple_II)
3. Anthony Faiola. The Design Enterprise: Rethinking the HCI Education Paradigm. *Design Issues*, Volume 23, Number 3 (Summer 2007), pp. 30-45
4. Adobe AIR (2007, Dec 5) Retrieved December 5, 2007 from <http://www.adobe.com/products/air/>
5. Brown, A., & Sugar, W. (2004). Integrating HCI into IDT: Charting the human computer interaction competencies necessary for instructional media production coursework. Association for Educational Communications and Technology; 27th, Chicago, IL, October 19-23, 2004
6. Browser Wars (2007, Dec 5). In *Wikipedia, The free encyclopedia*. Retrieved December 5, 2007, from [http://en.wikipedia.org/wiki/Browser\\_Wars](http://en.wikipedia.org/wiki/Browser_Wars)
7. Canny, J. (2006). The future of human-computer interaction. *Queue*, 4(6), 24-32.
8. Electronic Performance Support Systems (2007, Dec 6). In *Wikipedia, The free encyclopedia*. Retrieved December 6, 2007 from [http://en.wikipedia.org/wiki/Electronic\\_performance\\_support\\_systems](http://en.wikipedia.org/wiki/Electronic_performance_support_systems)

9. Hawthorn, D. (2007). Interface design and engagement with older people. *Behaviour & Information Technology*, 26(4), 333.
10. HyperCard. (2007, Dec 3). In *Wikipedia, The free encyclopedia*. Retrieved December 3, 2007 from <http://en.wikipedia.org/wiki/HyperCard>
11. Kolko, J. (2004). Mixing disciplines in anticipation of convergence: A curriculum for teaching interaction design to industrial designers. *Interactions*, 11(4), 18-23.
12. Moon, Y., & Nass, C. (1996). How "real" are computer personalities? Psychological responses to personality types in human-computer interaction. *Communication Research*, 23(6), 651-674.
13. Mozilla (2007, Dec 4). In *Wikipedia, The free encyclopedia*. Retrieved December 4, 2007 from <http://en.wikipedia.org/wiki/Mozilla>
14. Muuss, Mike. History of Computing Information. n.d. Retrieved December 3, 2007, from <http://ftp.arl.army.mil/~mike/comphist/>
15. Parush, A. (2006). Toward a common ground: Practice and research in HCI. *Interactions*, 13(6), 61-62.
16. Ross, J., & Schulz, R. (1999). Can computer-aided instruction accommodate all learners equally? *British Journal of Educational Technology*, 30(1), 5-24.
17. Visual Basic (2007, Dec 3). In *Wikipedia, The free encyclopedia*. Retrieved December 3, 2007, from [http://en.wikipedia.org/wiki/Visual\\_Basic](http://en.wikipedia.org/wiki/Visual_Basic)
18. Wenger, M. J. (1991). On the rhetorical contract in human-computer interaction. *Computers in Human Behavior*, 7(4), 245-262.

19. XULRunner – MDC (2007, Nov 24) Retrieved December 5, 2007 from

<http://developer.mozilla.org/en/docs/XULRunner>