# A Review of Multimodel User Interface in Human Computer Interface Design

**[1]Ying A. Chu and [2]Michael Wu Chu**
[1,2]Technology and Operations Management, Harvard Business School, Boston, MA 02163.
[1]chuaying@hbs.edu

**ArticleInfo**
**R. Arulmurugan et al. (eds.),** *First International Conference on Machines, Computing and Management Technologies*, Advances in Intelligent Systems and Technologies
Doi: https://doi.org/10.53759/aist/978-9914-9946-0-5_4

**Abstract** – The evolution of computers has inspired the conception of Human-Computer Interaction (HCI). Experiments in HCI are increasingly including members of the next generation (those in their twenties and thirties) who are well educated and proficient in modern technologies. In this paper, behavioral and technological issues are both addressed in the field of HCI. The primary goal of this research in HCI is to shed light on previously unseen perspectives of human behavior and its connection to technology. The concept of resilience has been used in several contexts, including the corporate world, psychology, education, and the economics. Self-efficacy, optimism, self-awareness, and the ability to derive meaning from one's experiences are just a few of the foundational factors that contribute to resilience. An improved organization's resilience may be achieved via the use of this method. All of this will be done by resilience in order to increase people's awareness of the tools at their disposal for dealing with the difficulties that already exist.

**Keywords –** Human-Computer Interaction (HCI), Graphical User Interfaces (GUIs), Computer-Human Interaction (CHI), Multi-Modal User Interfaces (MUI)

## I. INTRODUCTION

Technology advances at an ever-faster rate these days, and the computer industry is no exception. In addition, humans rely and interact heavily on computers for a wide range of functions. Human-Computer Interaction (HCI) define the manner in which users interact with computers as well as other technology to discover what more approaches may be created to facilitate effective cognitive interaction between humans and computers (as depicted in **Fig. 1**). HCI has four major components: the user (human), the goal-oriented task, the interface, and the context (see **Table 1**). HCI research is expanding beyond the improvement of interaction quality, and different authors have presented their definition of the term.

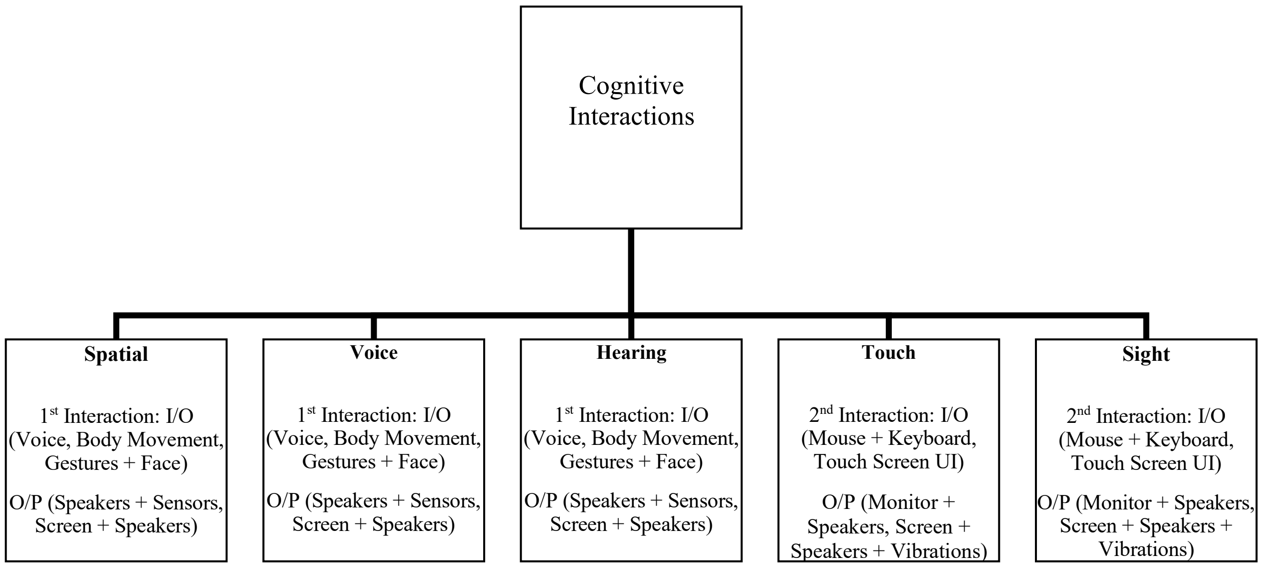

Cognitive Interactions
Spatial
1st Interaction: I/O (Voice, Body Movement, Gestures + Face)
O/P (Speakers + Sensors, Screen + Speakers)
Voice
1st Interaction: I/O (Voice, Body Movement, Gestures + Face)
O/P (Speakers + Sensors, Screen + Speakers)
Hearing
1st Interaction: I/O (Voice, Body Movement, Gestures + Face)
O/P (Speakers + Sensors, Screen + Speakers)
Touch
2nd Interaction: I/O (Mouse + Keyboard, Touch Screen UI)
O/P (Monitor + Speakers, Screen + Speakers + Vibrations)
Sight
2nd Interaction: I/O (Mouse + Keyboard, Touch Screen UI)
O/P (Monitor + Speakers, Screen + Speakers + Vibrations)

**Fig 1.** Cognitive interaction of HCI

Human-Computer Interaction (HCI) emerged in the late 1980s, at the height of the era of the personal computer when desktop computers first became common in businesses and homes [1]. The advancement of HCI began with word processors, calculators and electronic toys. However, computing became ubiquitous and all-powerful with the arrival of the internet and the advent of mobile devices and technologies such as voice-oriented IoT (Internet of Things). Interaction with users has developed due to different technological advancements. Therefore, there has been an increasing requirement for technologies to facilitate interactions between humans and machines that are increasingly natural. This diversifies HCI as a technological discipline, integrating earlier separate domains of research e.g., linguistics, cognitive engineering, and neuroscience. Currently, the main goal of HCI research is to create, test, and evaluate user-friendly software for computers and other electronic devices. It encompasses such sub-fields as user-based design, user experience design, and user-interface design.

Barricelli and Fogli [2] defined HCI as the research of how individuals interact with computer systems to accomplish a wide range of activities. How they are putting it to good use so that computer contact is something that people look forward to and benefit from. HCI was once limited to the realm of computing but has now broadened to include practically every aspect of IT architecture in use today. It encompasses a wide range of approaches to HCI that aim to streamline and improve productivity. The first step toward an intelligent HCI is the ability to react and sense correctly in accordance with human (the user's) emotional feedback and identify, understand the affective states displayed by the user instinctively. Human-computer interface design methods are also discussed in detail in this study. To understand how people use and react to interactive technology, researchers in the subject of HCI draw on a wide range of disciplines, including those of cognitive science, organizational and computer science, psychology, and the social sciences.

There are various ways in which humans and computers communicate, and each relies heavily on the interface between them. HCI goes under a few different names, including Computer-Human Interaction (CHI), Man-Machine Interaction (MMI), and Human-Machine Interaction (HMI). Modern Graphical User Interfaces (GUIs) are used in many different types of software, including desktop applications, mobile devices, computer kiosks, and web browsers. Multi-Modal User Interfaces (MUI) and GUI are on the rise, permitting individuals to interact with embodied element agents in the manner that were earlier impossible. The Voice User Interface (VUIs) is employed for synthesis systems and voice detection. As the domain of HCI has expanded, the quality of interactions has improved, and numerous new study avenues have opened up. Multimodal over unimodal, smart adaptive over action/command based, and active against passive are some of the themes that have been prioritized by the various research streams in lieu of the design of conventional interfaces.

Wu and Gu [3] defined HCI as "a field that is concerned with the design, assessment, implementation, and research of interactive computing systems for human use and the study of key phenomena surrounding them," according to the ACM. User happiness is a crucial part of HCI (or End-User Computer Satisfaction). Onward it goes, saying: "Since HCI is concerned with how people and computers communicate, it makes use of findings from studies of both human and technological communication. On the hardware side of things, methods from computer graphics, operating systems, programming languages, and development environments are all applicable. Communication theory, visual and industrial design, linguistics, the social sciences, cognitive and social psychology, and human aspects like computer user happiness are all related to the human side of things. Mechanical and architectural design principles are also important."

Successful HCI relies on the contributions of experts from many fields. Issues of all sorts may arise as a result of poorly conceived human-machine interactions. The nuclear meltdown catastrophe at Three Mile Island is a typical case in point; and studies into the incident ultimately revealed that flaws in the interaction between humans and machines contributed to the tragedy [4]. Similarly, incidents have occurred in the aviation industry when manufacturers deviated from the "conventional" layout for flight instruments or throttle quadrants. This was the case even if the new designs were thought to improve fundamental aspects of human-machine interaction. It follows that there were unforeseen consequences to the otherwise sound plan.

In this paper, we define Human-Computer Interaction (HCI) as the research of human computers and humans collaborate, specifically how well computing systems are created to collaborate with humans. HCI is currently researched by a significant number of large universities and organizations. With a few notable exceptions, the simplicity of computers has not always been a priority for those creating new systems. There are many people who use computers nowadays who believe that manufacturers do not put enough effort into making their devices "user-friendly." The demand gap for services that computer systems can provide has always surpassed the desire for ease-of-use, but the engineers of these systems may counter that computers are significantly complex items to produce and design.

One crucial aspect of HCI is that users' preconceptions and mental models of their interactions vary, as do their methods of acquiring and retaining information (various “cognitive formats” as in, for instance, "left-brained" and "right-brained" users). In addition, there is the influence of cultural and national diversity. Another thing to keep in mind while developing and researching HCI is that user interface technologies is developing at a rapid rate, paving way for novel options for interaction that may render past studies obsolete. Last but not least, as users become more proficient with modern interfaces, their tastes will naturally evolve. The remainder of the paper has been organized as follows: Section II focuses on a background analysis of the Human-Computer Interaction (HCI). Section III is about the iteration in the HCI design. In Section IV, a conclusion to the article is reached.

## II. BACKGROUND ANALYSIS OF HUMAN-COMPUTER INTERACTION

Human-Computer Interaction (HCI) was introduced in the 1980s with the advent of personal computers such as Commodore 64, IBM PC 5150, and Apple Macintosh. Word processors, gaming consoles, and accounting software were only some of the initial applications for the newly accessible high-tech electronic devices. Therefore, as computers evolved from room-sized, costly instruments exclusively created for experts in specialized environments, the necessity to develop

HCI, which was both effective and simple for less savvy users, grew in importance. Beginning with its roots in computer science, HCI eventually grew to include fields like cognitive science and human-factors engineering. Rapid scholarly interest in HCI sprang out after its introduction. HCI practitioners and researchers identified it as a critical tool in creating awareness that users and the computer devices should have interactions that are natural and unstructured. HCI experts at first prioritized desktop computer usability (i.e., experts focused their attention on the simplicity of computers). HCI integrates four major components: the user (human), the goal-oriented task, the interface, and the context (see **Table 1**).

**Table 1.** Key components of HCI

| Component | Description of the HCI components |
|---|---|
| **The user** | The term "user" is employed when describing an individual or collective group who take part in a particular activity. User requirements, aims, and behavior patterns are the focus of HCI research. It considers the users' knowledge, past experience, and emotions to establish frictionless interactions between a computer and the user. |
| **The goal-oriented task** | A user performs actions on a computer with the end result in mind. The computer's digital representation of physical items is used for this purpose. A mission for an aviation website may be, for instance, booking a flight to a certain place. The following considerations are important for enhancing the user experience in goal-oriented scenarios: (i) How complicated the user's intended job really is. Interaction with the digital item requires (ii) the appropriate background knowledge and expertise. (iii) How long it will take to perform the project |
| **The interface** | When properly designed, the interface is a vital part of HCI that may boost the usability of the whole system. There are several facets of an interface to think about, including the mode of interaction (touch, click, gesture, or speech), the screen's resolution, the size of the display, and the contrast between colors. Each user has the ability to fine-tune these to their own preferences. |
| **The context** | HCI does not only concern with making it easier for users to interact with computer devices, but also to consider the environment in which these devices are utilized. When creating a mobile app, designers must take into account a number of factors that might affect the app's performance or user experience, such as the time of day or the strength of the network connection. The quality of the product's interaction with its ultimate user may depend in large part on such details. |

*Humans*

Products of HCI are generated by humans and utilized by people; as a result, they are also referred to as "user products." Memory, focus, problem-solving, learning, motivation, physical abilities, conceptual models, and variety are all aspects of the human/user that may be analyzed in terms of their role as an information processor. Conversational specialized languages and linguistics are discussed, along with the ways in which people use language to engage with computers and one other.

*Computers*

While computers themselves serve primarily as a medium for human communication, a wide range of peripherals allows for their manipulation (the users). Computers provide a wide range of features, and their modular design means that any user may tailor their machine to their own requirements. Depending on the process of the system and the time required for that operation, computers can perform a wide range of tasks, including storing and retrieving information, measuring, easily counting, and performing a wide range of mathematical operations, responding swiftly to any query, processing or calculating data, and completing tasks that require repetition in a shorter amount of time.

*Interaction*

In each interaction, both parties will always gain something. Here, people may make use of various computer parts to engage in meaningful interaction with the machines. The user typically communicates with the computer in order to have their question answered. Here, the numerous distinctions between robots and humans become abundantly obvious. Interaction between humans and computers is essential to their productive collaboration. In order to create a system that is both effective and user-friendly, it is important to factor in what you know about computers and people, and to get feedback from a wide range of people throughout the design phase. The timeframe and the cost of development are two major factors to think about while creating a real-time system. All of us are essentially immersed in research into the human factors behind the system's conception, execution, and operation.

*User Interface*

The software project would be nothing without the user interface. Specifically, HCI Users and computers will be interacting with one another; thus, the user interface is also very important in computer interaction. The primary functions of a user interface (UI) are the management of inputs (what the user enters into the system) and the presentation of results. The developer side will take into account a number of factors in the context of providing better services to consumers at the time of developing the user interface. The services provided by an interface include making it simple for users to interact with it, ensuring their satisfaction once they do so, allowing for the possibility of mistake in real time, and maximizing the interface's effectiveness for those using it.

## III. ITERATION IN THE HUMAN-COMPUTER INTERFACE DESIGN

As a result of recent developments in the microelectronic sector, inexpensive and easily accessible high-tech equipment is now within reach for a far larger population [5]. However, the full advantages of these technologies can only be realized if they are also cognitively accessible. To do so, we may look at it from two perspectives: that of the professional programmer, who must be adept at constructing elaborate programs, and that of the average computer user. Using both this perspective and that of the end users, who are often a technological novice but expert in his or her own field of application. Since the two groups have varying degrees of comfort in dealing with technologically based issues, each scenario presents its own unique challenges for the field of Human Factors.

The primary focus of this study is on user experience issues related to ties. To begin, we assume that if technological-oriented tools are employed, it is because the user's issues are too large for the methods now in use. Consequently, the adoption of the new technology must resolve existing issues rather than spawn new ones. This assumption leads to the need that the new tool is user-centric, rather than user-centric (which is typically the case). This alludes to the ability of the system to effectively “fit” to the end users’ motor capabilities, problem-solving approaches, or the cognitive structure. But how can we systematically and methodically develop such a "fit"? Most of the relevant literature is included in the annotated bibliographies found in [6]. Despite these research efforts, it is clear that there is still a gap in the literature when it comes to giving the designers with the required metrics to allow them to predict the success of a single interface design over the other. The design of the user interface is still more of an "art" than a "science".

There are two paths open to the designer of the user interface when faced with a challenge for which there is no existing literature-based answer. One option is to adopt a scientific method, which involves conducting formal tests to help fill in the gaps in our understanding. A system that was normally required "yesterday" may be delivered using an engineering technique in which an ad hoc strategy is adopted. Despite the significance of seminal works of Willis, Chavkin, and Leung [7], the latter strategy is significantly more common because of practical considerations. In this work, we want to propose a methodology for design that, while on the one hand adopting a pragmatic engineering approach, also makes an effort to collect data that might serve as the foundation for a deeper scientific knowledge of the issue at hand. To illustrate our points, we'll use data from a case study in computer music, a field that's functioned as a kind of "test bed" for many of the ideas discussed here.

### *Iterative Design*

Understanding the user's behavior is crucial when designing a system to "fit" them. It is very improbable that the initial implementation of any user interface will operate as effectively as it might or should due to the limits of the current analytical techniques and our inabilities to effectively estimate the performance of the system in "real-world" context. In such a context, it is imperative to utilize an iterative strategy to design and try various things until what works best is found. There are, however, two issues that stand out right away. The first is how to determine whether something is "correct" and what makes it so. Second, how can one go about making such an iterative method both financially and practically viable?

Each design iteration in the outlined method is seen as a prototype whose function is to put a significant portion of the issue under scrutiny. Users who serve as "guinea pigs" are used to test and evaluate each prototype before it is put into production. The prototype's effectiveness is assessed in light of this data, and plans for the next version are made. As we circle back to the initial questions, we can see that the following three factors are crucial to the efficient use of the iterative approach: (i) specifically, what is it that has to be prototyped, and how will it be done? (ii) When and how do observations occur? (iii) In what ways are findings assessed, and then used? There is no easy answer to any of these problems; but, our years of experience have led us to feel that there are broad methods which can be taken in each of these areas, which, when considered together, make the iterative strategy practical in various contexts. A further analysis of these concerns will be discussed in the sections below.

### *Prototyping*

There is a direct connection between the dilemma of what to prototype and the practice of traditional problem-solving. The designer's job is to break down the complex challenge at hand into a series of simpler ones. There are some of them that need to be tested as prototypes before they can be dealt with effectively. Identifying a "critical mass" of the issue at hand allows the designer to zero down on the most important aspects of the challenge at hand and work as efficiently as possible. Furthermore, the designer must create and have access to "prototyping tools."

The development, deployment, and upkeep of application software are all simplified with the help of prototyping tools. They're the instruments without which iterative design would be hopeless. As far as we're concerned, the designer need to make every effort to shape his programming environment to the point where all viable solutions to any given challenge may be equally put to the test. By avoiding the "way of least resistance," users may lessen the impact of predetermined biases in the system. We believe that if a design has to be adopted due to the cost of testing a feasible alternative, then more effort should be put into improving prototyping tools rather than designing new prototypes. This section will continue with a more in-depth examination of such instruments and the many levels at which they operate.

*System Level Tools*

Insufficient emphasis has been placed on seeing the programming environment as a whole, which has often stymied the advancement of design initiatives. The "latest" technology is often prioritized above the systems' ability to run commonly used software, such as word processors, graphics packages (like GPAC), computer languages, and debugging tools. However, the viability of a system can only be determined by how well it accommodates experimental programming. Deutsch and Taft provide further in-depth coverage of this topic. We'll use one scenario—the impact of high-level languages on prototyping setup efficiency—to illustrate our points.

*High Level Language*

To be solved, most issues need complicated ideas to be prototyped. The ideas to be assessed and their interrelationships must be able to be expressed succinctly in the language used for economic encoding. For many of the issues that pique a Human Factors researcher's attention, there is no easy answer when it comes to choosing an appropriate high-level language in which to conduct the study. Consider the challenge of outlining the workflow and causal connections for a sophisticated real-time control task's user interface. When operating a motor vehicle, a human operator must coordinate the expressions of distinct data channels simultaneously. Nonetheless, in case the destinations of these streams of data are a digital computer, it is challenging to identity high-level programing languages, which provide a simplified and succinct explanation of how these sets of data are to effectively manage the activities on progress.

The aforementioned findings have varied repercussions. First, a key barrier to progress in computer-based Human Factors study is the use of improper specification/implementation languages. The researcher should give careful consideration to the linguistic differences that may arise. Carolina Spindola Rangel Dias, Rojas Soares, Jäschke, Bezerra de Souza Jr, and Pinto [8] note the significance of notation as a mental aid. Second, it's crucial to study languages that facilitate the automated monitoring of user activities and the execution of parallel operations, data-driven operations, graphical interactions, and real-time event scheduling.

*Application Level Support*

Creating a collection of well-defined modules, or "building blocks," that enable rapid prototyping and testing is the foundation of any successful prototyping environment. In many "real-world" situations, prototyping is constricted to smaller subset of issue categories, hence acknowledging this fact facilitates the deployment of this type of tool-building paradigm. There are some transaction types that come up often enough that they may be given their own high-level support (or "breadboarding") modules, making it easier for the programs programmer to design the system. As an example of the kind of work we've done developing such modules for use in computer music systems, we provide some here.

*Menu System*

The use of menus in HCIs is fundamental and ubiquitous. It is associated with two primary groups of difficulties. The first issue is with the visual style of the game. This entails the design and organization of the menu. The second issue is figuring out what leads to what while using the menu in different situations. These two activities slow down the prototype process and drain scarce design resources that might be put to better use solving challenges in the application domain if they were better supported by appropriate tools.

Given the importance of this issue, we have spent significant time and effort developing a system that may be used as a model for menu-based interaction and documenting the process along the way. The creator of menu-driven software may simply "plug-in" relevant values, and the system will take care of program startup, event monitoring, and command invocation on their behalf. An intriguing instance of a prototype environment "bootstrapping" itself is the menu system. Specifically, the package has been developed and improved in light of what has been seen and evaluated about the nature of menu-based conversation.

The present menu system solves the second of two issues with menu-based interaction (after the first layout difficulty). To do the job, we need to create a program that employs CAD principles in menu design, allowing the user to create the values that will be "plugged into" the current module's pre-existing template. As a result of this iterative process, we are able to efficiently prototype a wider range of concepts: the graphics package was written in a high-level language, the current menu system was written in a low-level language, the menu layout model will be written in the low-level language, and the application-level software will be written in a higher-level language. Each procedure aids the applications developer's capacity to do the assigned work. The most salient takeaway from the above is the need to use an iterative design process across both prototype creation and prototyping tool creation.

*Directory Windows*

The inability to locate previously designated data or files is a common issue among computer novices. There are a number of ways that this issue shows up in people's lives. To begin, the user faces challenges due to the ambiguity introduced by the potential of two coexisting copies of that data, one on the disk and the other in the main memory, after having learnt to correlate specific data with a unique file name. What kind of information is referred to during the retrieval process, for instance? Having to remember the name of the file, the spellings, isolating files of particular forms from among the different kinds, which may exist, type the names of files, and navigate between files is cumbersome and adds to the conceptual challenge.

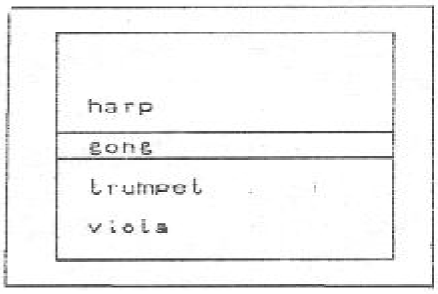
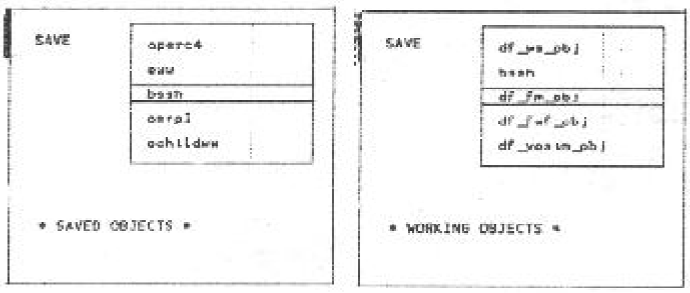


**Fig 2.** An example of a simplified directory window



(a) Retrieving the score from disks

(b) Retrieving the score from the main memory

**Fig 3.** Switches of the directory window

As a group, these issues have one thing in common: they force the user to devote attention and mental energy to something that is tangential to the primary purpose for which the computer was initially chosen. They are mostly technical issues that prevent the user from focusing on the issues that matter most to them. These issues affect a wide variety of domains, and they may be fixed for many different types of systems by using the same set of standard modular tools. The end result is a friendlier interface for the user and a collection of high-level building-blocks for the designer that makes it easier to create working prototypes of systems. The "directory windows" methodology we've developed for making these modules is the foundation of our work in this area. A directory window is a section of the display that displays and provides access to files of a certain format. The code, which supports these windows, is constructed with various well-documented modules, which provide the programmer controls over the windows' size and location on the screen, the kind of file it displays, an alternate method for browsing and choosing files, the origin and representations of the files in the window, and more. This section began with outlining some of the issues that might arise while attempting to retrieve a file; the examples that follow serve to demonstrate these ideas and tie them to these issues.

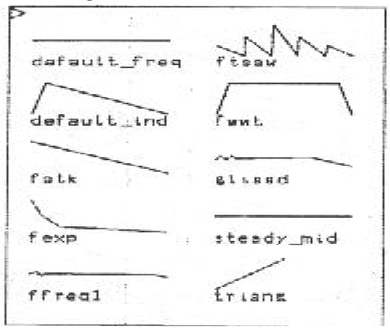


**Fig 4.** Iconic Representation of File Contents

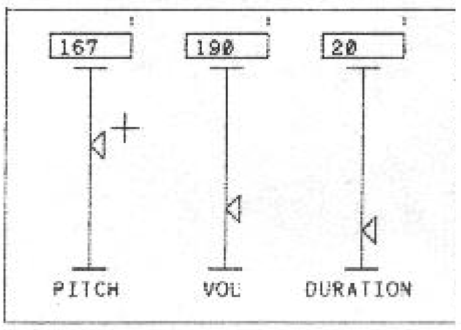


**Fig 5.** Simplified graphic potentiometer.

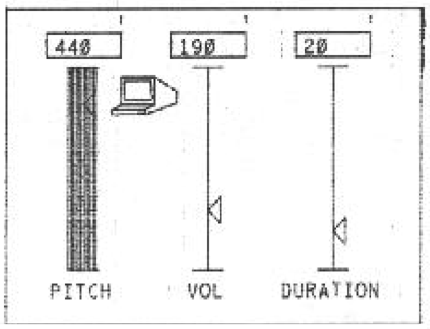


**Fig 6.** Observations of the property in minimizing operator errors

The bare minimum representation of the window is a rectangle composed on different names of files, one of which is located in the middle of a set of horizontal lines (see **Fig. 2**). All of the names correspond to actual files. The files whose names show in a given window are all, by definition, musical scores. The "current" file is the one whose name is in between the lines. If we imagine this in the aspect of the score editors, the names between two different lines would signify the working names of the scores. Keeping with the theme, the editor may be used to load new files by selecting them from a list that appears between the horizontal lines. It is possible to achieve this in a number of ways, such as simply pointing at the names in the window or by employing transducers to "scroll" different words in the window, hence allowing for accessibility of file names, which were earlier hidden. In addition to its versatility, this method is significant since it allows the user to quickly and easily access files without worrying about the correct spelling of their names or the hassle of entering them in. Since all files are keyed by type, the applications developer is also given a way to filter out information that isn't relevant to a certain scenario. Allowing two copies of a file to coexist, one on the disk and one in the main memory, creates ambiguity, which has been pointed to as a potential issue. An internally consistent approach is offered in the form of the "switch" shown in **Fig. 3** to deal with the issue.

In particular, as shown in **Fig. 3a**, when the window is set to display "Saved Scores," an explicit text is sent showing that the files being retrieved from the window are saved versions of the same file on the disk. Conversely, when the toggle is set to "Working Scores," as shown in **Fig. 3b**, users are given a visual cue to indicate that the files being seen in the window are just temporary working copies saved in main memory. The saved or functional version of a file may be retrieved with the flip of a switch, and the user can also use it to check whether the file has been saved.

One last thing has to do with directory windows, which allow for easy file browsing and allow for specific file types to be picked based on their contents rather than their (sometimes arbitrary) name. In **Fig. 4**, we can see that along with the filename, we are also given a graphical representation of the information contained inside the file (in this example, a set of time-varying functions). The last example highlighted that directory windows may be represented in a variety of ways while still preserving a consistent mode of interaction. This is a hunch that can and should be explored further in further studies. By using a suitable iconographic representations of file data, the user interface is widely developed in different practical scenarios, as we have discovered. This is why prototyping tools are so important; they make it easier to try out new ideas quickly.

*Graphic Potentiometers*

As a broad category, CAD/CAM/CAE applications include musical creation and performance. Setting or adjusting the parameters or scaler values, is one type of transaction that has been seen to be common in these applications. That is, any sort of exchange that requires manual intervention, such as twiddling a dial or punching in a code. As this is a recurrent issue, we've created a second prototype tool that, once again, makes it easy to "bread-board" model by providing highly-parameterized and well-documented building block components. These parts are known as graphic potentiometers. **Fig. 5** displays three graphic potentiometers.

To provide an analogue (high-low) representation of the current value, they use the orientation of a triangle to represent the "handle" of a physical potentiometer. It may be possible to change the value by moving the control up or down using a mouse or other point-and-click interface, or any other transducer a graphics software can accommodate. There are two effects of adjusting. Real-time updates are made to both the visual display and the potentiometer's related parameter. So, using the illustration as an example, if a tone were playing while the potentiometer of the pitch was being altered, the resultant aural transition would be detected.

The graphic potentiometer has only been discussed in traditional perspectives up to this point. Nonetheless, take note of the box above the potentiometer, which displays a numerical readout of the parameter's current value. This display is likewise dynamically updated in response to the user's changes to the parameter. In addition, the user may type in a value immediately by pointing to the box and clicking. **Fig. 6** depicts this phenomenon. This case serves to illustrate two key things. We might start with the fact that the "tracking sign," a "+" in the original illustration, has been replaced with a terminal icon. This serves as a visual cue to indicate that a value must be entered. Second, the inputted value fits exactly within the designated box as it is being written (instead of being placed at the bottom of the screen until they are finished, at which point they will be moved to their final position). Both functions have the advantageous quality of directing the user's visual attention to the precise spot on the screen where they need it.

The notion of prototyping tools is presented in this work. We have provided examples at both the systemic and the application levels. An iterative design process is encouraged, and this is why it is crucial to put effort into creating a good prototype environment. Having access to high-quality prototype modules also helps designers avoid getting bogged down in irrelevant details and instead focus on the core challenges at hand. Finally, it was noticed that the prototyping setting might be debugged in a manner that is cost-effective by employed an iterative approach of prototype, observe, and evaluate to prototyping approaches themselves.

*Observation*

Tools for gathering information to use in assessing interfaces are the focus of the iterative approach's second major component. Our discussion is based on the premise that prototypes of the to-be-evaluated interfaces exist and will be subjected to "real-world"-like testing. To restate, until better predictive and analytical frameworks are established, the relative usefulness of different interface designs can only be determined via observation of test participants completing typical tasks. Obviously, the nature of the test and the criteria for success play major roles in determining the viability of various observational methods. The rest of this section is dedicated to enumerating and characterizing the primary possibilities so that you may make an informed decision. This must be a very high-level debate because of time constraints.

There is a clear dichotomy between the two types of methods covered here: those that need the subject to actively participate by providing verbalizations, and those that do not. The first set is used to monitor subjects' actions in real time, but it's impossible to see the extensive amount of thought and expertise a user puts into operating a prototype. Here's when that bag of linguistic tricks comes into play. To get a sense of the user's decision-making process and the factors that influence it, it's important to learn as much as possible about the user without introducing too many distortions into their behavior from having to provide a verbal report. To the same extent, the report's own distortions as a representation of the user's behavior must be kept to a minimum.

*Non-Verbal Techniques*

*Observation by Researchers*

The researcher will stand above the participant and witness the exchange from a distance. Human observers may pick up on fundamental issues at the lowest levels of interaction that the user, focused on the complex cognitive aspects of the activity at hand, may miss. One example of the kind of knowledge that may be learned using this method is the proper manner to specify the number "1" while working with software. Researchers observing a user at work on a teletypewriter could detect that the user has a bias toward entering the letter "l" instead of the letter "a," based on their familiarity with standard typewriters. Since the subject may never report this happening, the researcher might not know it's happening until they see it for themselves. Regular use of this method, however, presents the logistical challenge of having researchers present at sessions at all times.

*Observation by Video Tapes*

Videotaping the session allows for a more permanent record that can be examined at the researcher's convenience. The costs associated with this method are rather evident. Further, it creates the difficulty of selecting what parts of the exchange to record. If the output and input transducers are dispersed across a wide region, it would be difficult to capture every facet of the interaction at once. Pope [9] at MIT have both employed this method. In the second, students were watched while they solved a musical issue by rearranging a group of tuned bells using this method. Children are filmed while they do the job, with the observer asking them questions about their thought processes and motivations. The child's cognitive structure that directed his behavior during the activity is investigated by reviewing the session's videotape and tape-recorded replies.

*Observation Systemy Systems*

Due to the fact that the system collects session data, monitoring by the system is very time-saving. The simplest form of this is a conversation between two people using a teletypewriter, which produces a word-for-word transcript of the conversation. The contact may be recorded (and perhaps time stamped) by modules included into the system for further study. Because they are used to aid in the creation of application software by offering a simple means of gathering requirements, these modules may be thought of as prototyping tools in and of themselves. As a downside, this method does not provide any means by which the kinesthetics of the exchange can be assessed.

The "Menu System" subheading provides an example of this method in action. The system may record the time, the menu item's name, and its position on the screen when they are selected. The software not only displays the user's hand movement graphically, but also keeps track of the sequence and time of menu item activation. There's no mistaking that this is a versatile graphics tool. It may reveal which parts of the menu get the most attention, allowing for more strategic placement of offerings. It may be used as a basis for the creation of further tools that "animate" or "play back" a user's session for the purposes of analysis. However, much has to be done before the full potential of the information this tool provides can be realized. For instance, it is not obvious that two menu items that get a lot of foot traffic together must always be placed near together.

*Verbal Techniques*

*Reports of Impression*

The Subject's Impression Report is a reflection on the session as a whole. Insights might be verbalized (to a tagging researcher or recording device) or written down. They are welcome before, during, or after the appointment. Taking some time to reflect on the session's progress may help the topic provide more up-to-date information. If it is uttered and documented, it might be synced with earlier-mentioned observation approaches. Any system feature that stores messages may be used to help with the comments if they need to be written. Date and time stamps may be added to communications in this situation for easier examination. Explicit disclosure by the user throughout the session might skew timing information for task performance and pull the user's focus away from the program. By writing down thoughts after a session, the user may reflect on their experience and see it more holistically, instead of as series of disconnected encounters.

*Detailed Session Commentary*

An in-depth session commentary is a subject's detailed account of what he did and how he was thinking during the session. The subject provides a running commentary—either verbally or in writing—on his goals, methods, and motivations. A researcher may use this sort of record to find instances when the interaction is less than ideal, even if the subject is oblivious to the notion that there is a problem. The prior method, which just recorded user-perceived difficulties and places of interest, is insufficient for detecting issues of this kind. Since each method yields distinct information, we clearly delineate between reports of impressions and extensive discussion. While the former draw attention to particular trouble spots, the later focus on the user's problem-solving methods in general. The quantity of information required for commentary is a substantial burden on the users in the entire session that should be considered when planning activities to be mentioned using this approached. There is also the option of showing the video recording of the session to the subject afterward and having them make remark on it. Last but not least, as Benefiel [10] stresses, it is important to use caution when extrapolating from such comments to conclusions about the user's expertise.

*Interrogation by the Researcher*
In order to reduce the likelihood of incorrect inferences being drawn from analysis of user reports, researchers might interrogate those involved in the process. There is no need to ask any questions at this time. Questions may be asked either before, during, or after the event, both orally and in writing. Properly posed questions may rapidly clear up any ambiguity between the researcher and the subject. Young [11] both employed questionnaires to investigate the contact with users. The benefits of this method include output that is both more organized and more amenable to analysis than the free-form remarks produced by the other unwritten methods. However, they are minimally able to adjust to changes in their environment. Methods have been described for gathering information on how a subject uses an interactive computer network to complete predetermined tasks. Interface assessment and comparison may begin with this setting and the functional prototypes of different user interface concepts. The most pressing open topic is how best to construct assessment criteria to direct the selection of research questions and methods.

*Evaluation*
The capacity to build efficient interfaces in a continuous and methodical manner is what we want to achieve. Our ability to statistically compare and assess designs is a necessity. We take a look at this in two different ways: first, by analyzing data gathered through benchmark testing in which participants carry out tasks on prototype computers; and second, by analyzing designs pre-implementation using models developed from experimentation. The iterative strategy is how we plan to carry out the former in order to reach the latter. If the experimental method can successfully create better models, then iteration may be avoided altogether, or at least devoted to more complex issues of the interface.

An notable case study of this strategy is done by Fridlund [12]. The research looked at several methods for pointing at and choosing text on a CRT. Russell [13] built on similar findings to create the "keystroke model," which has been shown to accurately predict user performance (although in a limited setting). However, it is still possible to argue that methods other from testing in real-world situations might be utilized to produce the findings to develop such models. There are, however, genuine risks that one must be aware of, so although this may be true in certain circumstances, it is not always the case. To begin, the time it takes for an action to be processed is a major factor in identifying the success of user interfaces. It is hard to conceive of many other methods that would adequately account for this variable. Second, research by Pernice and Budiu [14] and others has shown that users, even experts, are not always able to accurately estimate the "goodness" of simplified interfaces whenever shown the designs on paper. Subjective evaluations of designs are very unreliable, even when compared to objective metrics, even when testing are conducted on real-world systems. The work in [15] provides one such demonstration.

The goal of this research was to see how various map presentation options affected users' ability to locate the quickest route between two cities. Changes from one displayed map to another may be made in one of two ways: either in a discrete fashion, with 0–25% overlap, or in a continuous fashion, with 35% overlap. It's interesting to note that while most individuals favored the continuous option, it also consistently resulted in the longest solution time. Therefore, it is evident that it is necessary to not only construct properly defined criteria for gauging performance, but also to test designs under simulated real-world settings. This criterion was determined through research in [16]. The data is summarized in **Table 2** below:

**Table 2.** Summary data of the results

| Summary of the Results | |
|---|---|
| **Easy to Learn** | (i) user guides are unnecessary; (ii) system usage requires no specialized data processing expertise; |
| **Adaptability in Task Management** | (i) Messages from the system with varying degrees of information depending on the user's state; (ii) shorter approaches to complete tasks for skilled users; |
| **Time Factors** | (i) reaction times for various operations; (ii) time needed to complete a certain job; |
| **Relationship to User Expectations** | (i) comparable system behavior in comparable circumstances; (ii) feedback given so that the user may see the results of his input; |
| **Fault Tolerance** | (i) just a partial retyping effort is necessary if the preceding input was incorrect; (ii) typographical errors are accepted; |
| **Adequate Usability Problem** | (i) accepts freely structured command input; (ii) requires the user to conduct few administrative or maintenance tasks; |
| **Self-Descriptiveness** | (i) a dialogue's order and flow are always transparent; (ii) display of system operations that is organized clearly; |
| **User Control** | (i) process reversal is feasible without negative consequences; (ii) command language syntax is homogeneous; |

The preceding list is not exhaustive, and even if we were to learn how to quantitatively evaluate each element while looking at a specific design, we would still be unable to give useful resources to the designer of the user interface. To begin, the interaction between these variables is intricate in any realistic setting. We may either create a predictive framework that takes just one or two measurements as its foundation for design, or we can learn to assess the transactions inside a given interface as per interdependencies and corresponding weights of these criteria. Our lab's research demonstrates the significance of the latter. The goal was to analyze and classify several visual notational approaches for defining the time/pitch data of indivudual notes in a musical score

Each studied method produced varied outcomes when evaluated using the aforementioned criteria. In any case, it's important to stress that there is no such thing as a "optimal" answer that can be determined out of context. One is superior for CAI, whereas the other excels in composition. What are the best practices for analyzing an application's component transactions on the basis of their needs relative to the criterion, and then utilizing those findings to determine how to best design the user interface? Possibly, but in a very ad hoc way. Moreover, expanding into unfamiliar application domains would provide far less desirable outcomes without a well-structured paradigm for interface design. This leads to the second issue, which is a pragmatic one that this example illustrates. The capacity to understand why an interface works and apply that understanding to the design of future systems is limited, even if we can retrieve a quantitative measurment of how effectively it works. Another issue is the insuffient model, which has to be gradually solved by the iterative method.

## IV. CONCLUSION

Human-Computer Interface (HCI) is an multi-disciplinary topic, which concentrates on how users interact with computer systems in the entire design process. HCI has extended from its original focus on computers to include almost all aspects of IT product design. Rather than providing answers or findings, the focus of this study has been on identifying issues. We have zeroed in on the gap between what is known scientifically and what is utilized in practice for the human-computer interface in interaction design. Due to this, designers do not have access to reliable models that can be used to foretell and assess the effectiveness of proposed interface designs. A potential solution for improving these models is an iterative procedure. The technique relied heavily on individuals' willingness to take part in the tests as actors. As a result, three items were identified as essential to facilitate such testing as well as making the approach practically and economically viable: adequate prototyping tools, strategies of observation, and ways of assessment. After applying the method to our own work and that of others presented in the article, we conclude that it is a useful tool for expanding our comprehension of user interfaces, if only because it allows us to zero in on the specific areas where we have the biggest knowledge gaps.

## References

[1]. J. M. Carroll, "Human Computer Interaction - brief intro," The Interaction Design Foundation. [Online]. Doi:https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/human-computer-interaction-brief-intro. [Accessed: 02-Dec-2022].

[2]. B. R. Barricelli and D. Fogli, "Digital twins in human-computer interaction: A systematic review," Int. J. Hum. Comput. Interact., pp. 1–19, 2022.

[3]. J. Wu and Y. Gu, "Analysis of online classroom education on the learning patterns for college students using human-computer interaction," Int. J. Hum. Comput. Interact., pp. 1–12, 2022.

[4]. "Backgrounder on the Three Mile Island Accident," Nrc.gov. [Online]. Doi:https://www.nrc.gov/reading-rm/doc-collections/fact-sheets/3mile-isle.html. [Accessed: 02-Dec-2022].

[5]. W. T. Chen and A. Tseng, "Overview of recent developments in microelectronic packaging," in Advances in Electronic Materials and Packaging 2001 (Cat. No.01EX506), 2002.

[6]. A. F. B. Ck and J. Setiawan, "Analysis of user experience resource planning with User Experience Questionnaire Framework (case study: Universitas multimedia nusantara)," Journal of Multidisciplinary Issues, vol. 1, no. 2, pp. 42–61, 2021.

[7]. N. Willis, N. Chavkin, and P. Leung, "Finding 'health' and 'meaning' in Texas-sized turnover: Application of seminal management principles for administration and research in U.s. public child welfare agencies," Adv. Soc. Work, vol. 17, no. 2, p. 116, 2017.

[8]. A. Carolina Spindola Rangel Dias, F. Rojas Soares, J. Jäschke, M. Bezerra de Souza Jr, and J. C. Pinto, "Extracting valuable information from big data for machine learning control: An application for a gas lift process," Processes (Basel), vol. 7, no. 5, p. 252, 2019.

[9]. S. B. Pope, "A Monte Carlo Method For The Pdf Equations Of Turbulent Flo," Mit.edu. [Online]. Doi:https://dspace.mit.edu/bitstream /handle/1721.1/60525/EL_TR_1980_012.pdf?sequence=1. [Accessed: 02-Dec-2022].

[10]. D. J. Benefiel, "Use caution when extrapolating from a small sample size to the general population," Anesthesiology, vol. 70, no. 1, pp. 160–161, 1989.

[11]. T. J. Young, "Questionnaires and Surveys," in Research Methods in Intercultural Communication, Hoboken, NJ, USA: John Wiley & Sons, Inc., 2015, pp. 163–180.

[12]. B. Fridlund, "The case study as a research strategy," Scand. J. Caring Sci., vol. 11, no. 1, pp. 3–4, 1997.

[13]. J. Russell, Ed., Keystroke-Level Model. Book on Demand, 2012.

[14]. K. Pernice and R. Budiu, "Hamburger menus and hidden navigation hurt UX metrics," Nielsen Norman Group. [Online]. Doi:https://www.nngroup.com/articles/hamburger-menus/. [Accessed: 02-Dec-2022].

[15]. "Subjective or objective performance evaluations?" Managementexchange.com. [Online]. Doi:https://www.managementexchange.com/hack/ subjective-versus-objective-performance-evaluations. [Accessed: 02-Dec-2022].

[16]. G. Ara, T. Cucinotta, and A. Mascitti, "Simulating execution time and power consumption of real-time tasks on embedded platforms," in Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing, 2022.