Human–Computer Interaction & Usability Engineering

Hailie Welker
Jonas Dickens

First Edition, 2012

ISBN 978-81-323-0937-6

Published by:
Academic Studio
4735/22 Prakashdeep Bldg,
Ansari Road, Darya Ganj,
Delhi - 110002
Email: info@wtbooks.com

Table of Contents

Chapter 1

Human–computer Interaction

A mouse is a pointing device that functions by detecting two-dimensional motion relative to its supporting surface.

Human–computer interaction (**HCI**) is the study, planning and design of the interaction between people (users) and computers. It is often regarded as the intersection of computer science, behavioral sciences, design and several other fields of study. Interaction between users and computers occurs at the user interface (or simply *interface*), which includes both software and hardware; for example, characters or objects displayed by software on a personal computer's monitor, input received from users via hardware peripherals such as keyboards and mice, and other user interactions with large-scale computerized systems such as aircraft and power plants. The Association for Computing Machinery defines human-computer interaction as "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." An important facet of HCI is the securing of user satisfaction.

Because human-computer interaction studies a human and a machine in conjunction, it draws from supporting knowledge on both the machine and the human side. On the machine side, techniques in computer graphics, operating systems, programming languages, and development environments are relevant. On the human side, communication theory, graphic and industrial design disciplines, linguistics, social sciences, cognitive psychology, and human factors are relevant. Engineering and design methods are also relevant. Due to the multidisciplinary nature of HCI, people with different backgrounds contribute to its success. HCI is also sometimes referred to as **man–machine interaction (MMI)** or **computer–human interaction (CHI)**.

Attention to human-machine interaction is important, because poorly designed human-machine interfaces can lead to many unexpected problems. A classic example of this is the Three Mile Island accident where investigations concluded that the design of the human-machine interface was at least partially responsible for the disaster. Similarly, accidents in aviation have resulted from manufacturers' decisions to use non-standard flight instrument and/or throttle quadrant layouts: even though the new designs were proposed to be superior in regards to basic human-machine interaction, pilots had already ingrained the "standard" layout and thus the conceptually good idea actually had undesirable results.

Goals

A basic goal of HCI is to improve the interactions between users and computers by making computers more usable and receptive to the user's needs. Specifically, HCI is concerned with:

- methodologies and processes for designing interfaces (i.e., given a task and a class of users, design the best possible interface within given constraints, optimizing for a desired property such as learnability or efficiency of use)
- methods for implementing interfaces (e.g. software toolkits and libraries; efficient algorithms)
- techniques for evaluating and comparing interfaces
- developing new interfaces and interaction techniques
- developing descriptive and predictive models and theories of interaction

A long term goal of HCI is to design systems that minimize the barrier between the human's cognitive model of what they want to accomplish and the computer's understanding of the user's task.

Professional practitioners in HCI are usually designers concerned with the practical application of design methodologies to real-world problems. Their work often revolves around designing graphical user interfaces and web interfaces.

Researchers in HCI are interested in developing new design methodologies, experimenting with new hardware devices, prototyping new software systems, exploring new paradigms for interaction, and developing models and theories of interaction.

Differences with related fields

HCI differs from human factors in that with HCI the focus is more on users working specifically with computers, rather than other kinds of machines or designed artifacts. There is also a focus in HCI on how to implement the computer software and hardware mechanisms to support human-computer interaction. Thus, human factors is a broader term; HCI could be described as the human factors of computers - although some experts try to differentiate these areas.

According to some experts, HCI also differs from ergonomics in that there is less of a focus on repetitive work-oriented tasks and procedures, and much less emphasis on physical stress and the physical form or industrial design of the user interface, such as keyboards and mice. However, this does not take a full account of ergonomics, the oldest areas of which were mentioned above, but which more recently has gained a much broader focus (equivalent to human factors). Cognitive ergonomics, for example, is a part of ergonomics, of which *software ergonomics* (an older term, essentially the same as HCI) is a part.

Three areas of study have substantial overlap with HCI even as the focus of inquiry shifts. In the study of personal information management (PIM), human interactions with the computer are placed in a larger informational context - people may work with many forms of information, some computer-based, many not (e.g., whiteboards, notebooks, sticky notes, refrigerator magnets) in order to understand and effect desired changes in their world. In computer supported cooperative work (CSCW), emphasis is placed on the use of computing systems in support of the collaborative work of a group of people. The principles of human interaction management (HIM) extend the scope of CSCW to an organizational level and can be implemented without use of computer systems.

Design principles

When evaluating a current user interface, or designing a new user interface, it is important to keep in mind the following experimental design principles:

- Early focus on user(s) and task(s): Establish how many users are needed to perform the task(s) and determine who the appropriate users should be; someone who has never used the interface, and will not use the interface in the future, is most likely not a valid user. In addition, define the task(s) the users will be performing and how often the task(s) need to be performed.
- Empirical measurement: Test the interface early on with real users who come in contact with the interface on an everyday basis. Keep in mind that results may be altered if the performance level of the user is not an accurate depiction of the real human-computer interaction. Establish quantitative usability specifics such as: the number of users performing the task(s), the time to complete the task(s), and the number of errors made during the task(s).
- Iterative design: After determining the users, tasks, and empirical measurements to include, perform the following iterative design steps:

1. Design the user interface
2. Test
3. Analyze results
4. Repeat

Repeat the iterative design process until a sensible, user-friendly interface is created.

Design methodologies

A number of diverse methodologies outlining techniques for human–computer interaction design have emerged since the rise of the field in the 1980s. Most design methodologies stem from a model for how users, designers, and technical systems interact. Early methodologies, for example, treated users' cognitive processes as predictable and quantifiable and encouraged design practitioners to look to cognitive science results in areas such as memory and attention when designing user interfaces. Modern models tend to focus on a constant feedback and conversation between users, designers, and engineers and push for technical systems to be wrapped around the types of experiences users want to have, rather than wrapping user experience around a completed system.

- **User-centered design**: user-centered design (UCD) is a modern, widely practiced design philosophy rooted in the idea that users must take center-stage in the design of any computer system. Users, designers and technical practitioners work together to articulate the wants, needs and limitations of the user and create a system that addresses these elements. Often, user-centered design projects are informed by ethnographic studies of the environments in which users will be interacting with the system. This practice is similar but not identical to Participatory Design, which emphasizes the possibility for end-users to contribute actively through shared design sessions and workshops.

- **Principles of User Interface Design**: these are seven principles that may be considered at any time during the design of a user interface in any order, namely

Tolerance, Simplicity, Visibility, Affordance, Consistency, Structure and Feedback.

Display designs

Displays are human-made artifacts designed to support the perception of relevant system variables and to facilitate further processing of that information. Before a display is designed, the task that the display is intended to support must be defined (e.g. navigating, controlling, decision making, learning, entertaining, etc.). A user or operator must be able to process whatever information that a system generates and displays; therefore, the information must be displayed according to principles in a manner that will support perception, situation awareness, and understanding.

Thirteen principles of display design

Christopher Wickens et al. defined 13 principles of display design in their book *An Introduction to Human Factors Engineering*.

These principles of human perception and information processing can be utilized to create an effective display design. A reduction in errors, a reduction in required training time, an increase in efficiency, and an increase in user satisfaction are a few of the many potential benefits that can be achieved through utilization of these principles.

Certain principles may not be applicable to different displays or situations. Some principles may seem to be conflicting, and there is no simple solution to say that one principle is more important than another. The principles may be tailored to a specific design or situation. Striking a functional balance among the principles is critical for an effective design.

Perceptual principles

1. Make displays legible (or audible). A display's legibility is critical and necessary for designing a usable display. If the characters or objects being displayed cannot be discernible, then the operator cannot effectively make use of them.

2. Avoid absolute judgment limits. Do not ask the user to determine the level of a variable on the basis of a single sensory variable (e.g. color, size, loudness). These sensory variables can contain many possible levels.

3. Top-down processing. Signals are likely perceived and interpreted in accordance with what is expected based on a user's past experience. If a signal is presented contrary to the user's expectation, more physical evidence of that signal may need to be presented to assure that it is understood correctly.

4. Redundancy gain. If a signal is presented more than once, it is more likely that it will be understood correctly. This can be done by presenting the signal in alternative physical forms (e.g. color and shape, voice and print, etc.), as redundancy does not imply repetition. A traffic light is a good example of redundancy, as color and position are redundant.

5. Similarity causes confusion: Use discriminable elements. Signals that appear to be similar will likely be confused. The ratio of similar features to different features causes signals to be similar. For example, A423B9 is more similar to A423B8 than 92 is to 93. Unnecessary similar features should be removed and dissimilar features should be highlighted.

Mental Model Principles

6. Principle of pictorial realism. A display should look like the variable that it represents (e.g. high temperature on a thermometer shown as a higher vertical level). If there are multiple elements, they can be configured in a manner that looks like it would in the represented environment.

7. Principle of the moving part. Moving elements should move in a pattern and direction compatible with the user's mental model of how it actually moves in the system. For example, the moving element on an altimeter should move upward with increasing altitude.

Principles Based on Attention

8. Minimizing information access cost. When the user's attention is diverted from one location to another to access necessary information, there is an associated cost in time or effort. A display design should minimize this cost by allowing for frequently accessed sources to be located at the nearest possible position. However, adequate legibility should not be sacrificed to reduce this cost.

9. Proximity compatibility principle. Divided attention between two information sources may be necessary for the completion of one task. These sources must be mentally integrated and are defined to have close mental proximity. Information access costs should be low, which can be achieved in many ways (e.g. proximity, linkage by common colors, patterns, shapes, etc.). However, close display proximity can be harmful by causing too much clutter.

10. Principle of multiple resources. A user can more easily process information across different resources. For example, visual and auditory information can be presented simultaneously rather than presenting all visual or all auditory information.

Memory Principles

11. Replace memory with visual information: knowledge in the world. A user should not need to retain important information solely in working memory or to retrieve it from long-term memory. A menu, checklist, or another display can aid the user by easing the use of their memory. However, the use of memory may sometimes benefit the user by eliminating the need to reference some type of knowledge in the world (e.g. an expert computer operator would rather use direct commands from memory than refer to a manual). The use of knowledge in a user's head and knowledge in the world must be balanced for an effective design.

12. Principle of predictive aiding. Proactive actions are usually more effective than reactive actions. A display should attempt to eliminate resource-demanding cognitive tasks and replace them with simpler perceptual tasks to reduce the use of the user's mental resources. This will allow the user to not only focus on current conditions, but also think about possible future conditions. An example of a predictive aid is a road sign displaying the distance from a certain destination.

13. Principle of consistency. Old habits from other displays will easily transfer to support processing of new displays if they are designed in a consistent manner. A user's long-term memory will trigger actions that are expected to be appropriate. A design must accept this fact and utilize consistency among different displays.

Human–computer interface

The human–computer interface can be described as the point of communication between the human user and the computer. The flow of information between the human and computer is defined as the loop of interaction. The loop of interaction has several aspects to it including:

- **Task Environment**: The conditions and goals set upon the user.
- **Machine Environment**: The environment that the computer is connected to, i.e. a laptop in a college student's dorm room.
- **Areas of the Interface**: Non-overlapping areas involve processes of the human and computer not pertaining to their interaction. Meanwhile, the overlapping areas only concern themselves with the processes pertaining to their interaction.
- **Input Flow**: The flow of information that begins in the task environment, when the user has some task that requires using their computer.
- **Output**: The flow of information that originates in the machine environment.
- **Feedback**: Loops through the interface that evaluate, moderate, and confirm processes as they pass from the human through the interface to the computer and back.

Current research

Hot topìcs in HCI include:

Group interfaces

Interfaces to allow groups of people to coordinate are common (e.g., for meetings, engineering projects, or authoring joint documents). The Internet has produced a major impact on the nature of organizations and on the division of labor. Current research studies models of the group design process to create systems with increased rationalization of design to help group work.

User Tailorability

End-user development studies how ordinary users could routinely tailor applications to their own needs and use this power to invent new applications based on their understanding of their own domains. Users, with their deeper knowledge of their own knowledge domains, could increasingly be important sources of new applications at the expense of generic systems programmers (with systems expertise but low domain expertise).

Embedded computation

Computation is passing beyond desktop computers into every object for which uses can be found. Embedded systems make the environment alive with little computations and automated processes, from computerized cooking appliances to lighting and plumbing fixtures to window blinds to automobile braking systems to greeting cards. To some extent, this development is already taking place. The expected difference in the future is the addition of networked communications that will allow many of these embedded computations to coordinate with each other and with the user. Human interfaces to these embedded devices will in many cases be very different from those appropriate to workstations.

Augmented reality

A common staple of science fiction, augmented reality refers to the notion of layering relevant information into our vision of the world. Existing projects show real-time statistics to users performing difficult tasks, such as manufacturing. Future work might include augmenting our social interactions by providing additional information about those we converse with.

Factors of change

The means by which humans interact with computers continues to evolve rapidly. Human–computer interaction is affected by the forces shaping the nature of future computing. These forces include:

- Decreasing hardware costs leading to larger memory and faster systems
- Miniaturization of hardware leading to portability
- Reduction in power requirements leading to portability
- New display technologies leading to the packaging of computational devices in new forms
- Specialized hardware leading to new functions
- Increased development of network communication and distributed computing

- Increasingly widespread use of computers, especially by people who are outside of the computing profession
- Increasing innovation in input techniques (i.e., voice, gesture, pen), combined with lowering cost, leading to rapid computerization by people previously left out of the "computer revolution."
- Wider social concerns leading to improved access to computers by currently disadvantaged groups

The future for HCI, based on current promising research, is expected to include the following characteristics:

- **Ubiquitous communication**. Computers are expected to communicate through high speed local networks, nationally over wide-area networks, and portably via infrared, ultrasonic, cellular, and other technologies. Data and computational services will be portably accessible from many if not most locations to which a user travels.

- **High functionality systems**. Systems can have large numbers of functions associated with them. There are so many systems that most users, technical or non-technical, do not have time to learn them in the traditional way (e.g., through thick manuals).

- **Mass availability of computer graphics**. Computer graphics capabilities such as image processing, graphics transformations, rendering, and interactive animation are becoming widespread as inexpensive chips become available for inclusion in general workstations and mobile devices.

- **Mixed media**. Commercial systems can handle images, voice, sounds, video, text, formatted data. These are exchangeable over communication links among users. The separate worlds of consumer electronics (e.g., stereo sets, VCRs, televisions) and computers are partially merging. Computer and print worlds are expected to cross-assimilate each other.

- **High-bandwidth interaction**. The rate at which humans and machines interact is expected to increase substantially due to the changes in speed, computer graphics, new media, and new input/output devices. This can lead to some qualitatively different interfaces, such as virtual reality or computational video.

- **Large and thin displays**. New display technologies are finally maturing, enabling very large displays and displays that are thin, lightweight, and low in power consumption. This is having large effects on portability and will likely enable the development of paper-like, pen-based computer interaction systems very different in feel from desktop workstations of the present.

- **Information Utilities**. Public information utilities (such as home banking and shopping) and specialized industry services (e.g., weather for pilots) are expected to proliferate. The rate of proliferation can accelerate with the introduction of high-bandwidth interaction and the improvement in quality of interfaces.

Academic conferences

One of the top academic conferences for new research in human-computer interaction, especially within computer science, is the annually held ACM's Conference on Human Factors in Computing Systems, usually referred to by its short name CHI (pronounced *kai*, or *khai*). CHI is organized by ACM SIGCHI Special Interest Group on Computer–Human Interaction. CHI is a large, highly competitive conference, with thousands of attendants, and is quite broad in scope.

Chapter 2

Computer Accessibility and Computer User Satisfaction

Computer accessibility

In human-computer interaction, **computer accessibility** (also known as *Accessible computing*) refers to the accessibility of a computer system to all people, regardless of disability or severity of impairment. It is largely a software concern; when software, hardware, or a combination of hardware and software, is used to enable use of a computer by a person with a disability or impairment, this is known as Assistive Technology.

There are numerous types of impairment that impact computer use. These include:

- Cognitive impairments and learning disabilities, such as dyslexia, ADHD or autism.
- Visual impairment such as low-vision, complete or partial blindness, and color blindness.
- Hearing impairment including deafness or hard of hearing.
- Motor or dexterity impairment such as paralysis, cerebral palsy, or carpal tunnel syndrome and repetitive strain injury.

These impairments can present themselves with variable severity; they may be acquired from disease, trauma or may be congenital or degenerative in nature.

Accessibility is often abbreviated to the numeronym **a11y**, where the number 11 refers to the number of letters omitted. This parallels the abbreviations of **internationalization** and **localization** as i18n and l10n respectively.

Special needs assessment

People wishing to overcome an impairment in order to be able to use a computer comfortably and productively may need a "special needs assessment" by an assistive technology consultant (such as an occupational therapist or clinical scientist) to help them identify and configure appropriate assistive hardware and software.

Where a disabled person is unable to leave their own home, it may be possible to assess them remotely using remote desktop software and a webcam. The assessor logs on to the client's computer via a broadband Internet connection. The assessor then remotely makes accessibility adjustments to the client's computer where necessary and is also able to observe how they use their computer.

Considerations for specific impairments

Cognitive impairments and illiteracy

The biggest challenge in computer accessibility is to make resources accessible to people with cognitive disabilities - particularly those with poor communication skills - and those without reading skills.

Their further development relies on public domain icons being available. Many people with a learning disability learn and rely on proprietary symbols. They thus become tied to particular products. The copyright owners are generally unwilling to release them on the web.

Other examples include Web accessibility a set of guidelines and two accessible web portals designed for people developing reading skills are peepo.com — try typing a letter with your keyboard for more — and peepo.co.uk with enhanced graphics, unique style controls and improved interactivity (requires an SVG supported browser).

Visual impairment

Another significant challenge in computer accessibility is to make software usable by people with visual impairment, since computer interfaces often solicit input visually and provide visual feedback in response. For individuals with mild to medium vision impairment, it is helpful to use large fonts, high DPI displays, high-contrast themes and icons supplemented with auditory feedback and screen magnifying software.

In the case of severe vision impairment such as blindness, screen reader software that provides feedback via text to speech or a refreshable braille display is a necessary accommodation for interaction with a computer.

About 8% of people, mostly males, suffer from some form of colour-blindness. In a well-designed user interface, colour should not be the only way of distinguishing between different pieces of information. However, the only colour combinations that matter are those that people with a deficiency might confuse, which generally means red and green and blue and green.

Motor and dexterity impairments

Some people may not be able to use a conventional input device, such as the mouse or the keyboard. Therefore it is important for software functions to be accessible using both devices; ideally, software uses a generic input API that permits the use even of highly specialized devices unheard of at the time of software development. Keyboard shortcuts and mouse gestures are ways to achieve this. More specialized solutions like on-screen software keyboards and alternate input devices like switches, joysticks and trackballs are also available. Speech recognition technology is also a compelling and suitable alternative to conventional keyboard and mouse input as it simply requires a commonly available audio headset.

The astrophysicist Stephen Hawking is a famous example of a person suffering from motor disability. He uses a switch, combined with special software, that allows him to control his wheelchair-mounted computer using his remaining small movement ability. This performs as a normal computer, allowing him to research and produce his written work, and as an Augmentative and Alternative Communication and environmental control device.

Hearing impairment

While sound user interfaces have a secondary role in common desktop computing, usually limited to system sounds as feedback, software producers take into account people who can't hear, either for personal disability, noisy environments, silence requirements or lack of sound hardware. Such system sounds like beeps can be substituted or supplemented with visual notifications and captioned text (akin to closed captions).

Software accessibility

Software API exist to allow assistive technology products (like screen readers, Text-to-speech, etc.) to work with softwares. The current or past APIs are:

- Microsoft Active Accessibility (MSAA) on Microsoft Windows
- Microsoft UI Automation on Microsoft Windows, replacing MSAA
- IAccessible2 on Microsoft Windows, a competitor of Microsoft UI Automation also replacing MSAA
- AT-SPI on UNIX and Linux
- Mac OS X Accessibility
- Java Accessiblity and the Java Access Bridge for Java software.

Web accessibility

Enabling access to Web content for all users is the concern of the Web accessibility movement. Websites can be designed to be more accessible by their conformance to certain design principles.

Screen readers are of limited use when reading text from websites designed without consideration to accessibility; this can be due to the differences between spoken and written language and the complexity of text, but it is mainly due to poor page design practices. The tendency to indicate semantic meaning using methods that are purely presentational (e.g. larger or smaller font sizes, using different font colors, or images or multimedia to provide information) restricts meaningful access to some users. Therefore designing sites in accordance with Web accessibility principles helps enable meaningful access for all users.

For example, web designers can ensure that navigation and content is as plain and simple as appropriate and long texts should provide summaries.

Computer user satisfaction

Computer user satisfaction (and closely related concepts such as *System Satisfaction*, *User Satisfaction*, *Computer System Satisfaction*, *End User Computing Satisfaction*) deals with user attitudes to computer systems in the context of their environments. In a broader sense, the definition can be extended to user satisfaction with any computer-based electronic appliance. The term *Computer User Satisfaction* is abbreviated to *user satisfaction* here. In general, the development of techniques for defining and measuring user satisfaction have been ad hoc and open to question. However, according to key scholars such as DeLone and McLean (2002), user satisfaction is a key measure of computer system success, if not in fact synonymous with it. Interest in the concept thus continues. Scholars distinguish between user satisfaction and usability as part of Human-Computer Interaction.

The Computer User Satisfaction Instrument and the User Information Satisfaction Short-form

Bailey and Pearson’s (1983) 39-Factor *Computer User Satisfaction (CUS)* instrument and its derivative, the *User Information Satisfaction (UIS)* short-form of Baroudi, Olson and Ives are typical of instruments which one might term as 'factor-based'. They consist of lists of factors, each of which the respondent then is asked to rate on one or more multiple point scales. Bailey and Pearson’s CUS asked for five ratings for each of 39 factors. The first four scales were for quality ratings and the fifth was an importance rating. From the fifth rating of each factor, they found that their sample of users rated as most important: *accuracy*, *reliability*, *timeliness*, *relevancy* and *confidence in the system*. The factors of least importance were found to be *feelings of control*, *volume of output*, *vendor support*, *degree of training*, and *organisational position of EDP* (the electronic data processing, or computing department). However, the CUS requires 39 x 5 = 195 individual seven-point scale responses. Ives, Olson and Baroudi (1983), amongst others, thought that so many responses could result in errors of attrition, caused by the increasing carelessness of the respondent as they fill in a long questionnaire. They thus developed the UIS. This only requires the respondent to rate 13 factors, and so remains in significant use at the present time. Two seven-point scales are provided per factor (each for a quality), requiring 26 individual responses in all.

The problem with the dating of factors

An early criticism of these measures was that the factors date as computer technology evolves and changes. This suggested the need for updates and led to a sequence of other factor-based instruments. Doll and Torkzadeh (1988), for example, produced a factor-based instrument for a new type of user emerging at the time, called an end-user. They identified end-users as users who tend to interact with a computer interface only, while previously users interacted with developers and operational staff as well. McKinney, Yoon and Zahedi (2002) developed a model and instruments for measuring web-customer satisfaction during the information phase. Cheung and Lee (2005) in their development of an instrument to measure user satisfaction with e-portals, based their instrument on that of McKinney, Yoon and Zahedi (2002), which in turn was based primarily on instruments from prior studies.

No rigorous definition of the term user satisfaction

As none of the instruments in common use really rigorously define their construct of user satisfaction, some scholars such as Cheyney, Mann and Amoroso (1986) have called for more research on the factors which influence the success of end-user computing. Little subsequent effort which sheds new light on the matter exists, however. All factor-based instruments run the risk of including factors irrelevant to the respondent, while omitting some that may be highly significant to him/her. Needless to say, this is further exacerbated by the ongoing changes in information technology.

In the literature there are two definitions for user satisfaction, ‘User satisfaction’ and ‘User Information Satisfaction’ are used interchangeably. According to Doll and Torkzadeh (1988) ‘user satisfaction’ is defined as the opinion of the user about a specific computer application, which they use. Ives et al. (1983) defined ‘User Information Satisfaction’ as “the extent to which users believe the information system available to them meets their information requirements.” Other terms for User Information Satisfaction are “system acceptance” (Igersheim, 1976), “perceived usefulness” (Larcker and Lessig, 1980), “MIS appreciation” (Swanson, 1974) and “feelings about information system” (Maish, 1979). Ang en Koh (1997) have described user information satisfaction (UIS) as “a perceptual or subjective measure of system success”. This means that user information satisfaction will differ in meaning and significance from person to person. In other words, users who are equally satisfied with the same system according to one definition and measure may not be equally satisfied according to another.

Several studies have investigated whether or not certain factors influence the UIS; for example, those by Yaverbaum (1988) and Ang and Soh (1997). Yaverbaum's (1988) study found that people who use their computer irregularly tend to be more satisfied than regular users. Ang en Soh's(1997)research, on the other hand, could find no evidence that computer background affects UIS.

A lack of theoretical underpinning

Another difficulty with most of these instruments is their lack of theoretical underpinning by psychological or managerial theory. Exceptions to this were the model of web site design success developed by Zhang and von Dran (2000), and a measure of user satisfaction with e-portals, developed by Cheung and Lee (2005). Both of these models drew upon Herzberg's two-factor theory of motivation. Consequently, their factors were designed to measure both 'satisfiers' and 'hygiene factors'. However, Herzberg's theory itself is criticized for failing to distinguish adequately between the terms *motivation*, *job motivation*, *job satisfaction*, and so on.

Future developments

Currently, some scholars and practitioners are experimenting with other measurement methods and further refinements of the definition for *satisfaction* and *user satisfaction*. Others are replacing structured questionnaires by unstructured ones, where the respondent is asked simply to write down or dictate all the factors about a system which either satisfies or dissatisfies them. One problem with this approach, however, is that the instruments tend not to yield quantitative results, making comparisons and statistical analysis difficult. Also, if scholars cannot agree on the precise meaning of the term *satisfaction*, respondents will be highly unlikely to respond consistently to such instruments. Some newer instruments contain a mix of structured and unstructured items.

Chapter 3

User Interface and 3D Interaction

User interface

In the industrial design field of human-machine interaction, the user interface is (a place) where interaction between humans and machines occurs. The goal of interaction between a human and a machine at the user interface is effective operation and control of the machine, and feedback from the machine which aids the operator in making operational decisions. Examples of this broad concept of user interfaces include the interactive aspects of computer operating systems, hand tools, heavy machinery operator controls. and process controls. The design considerations applicable when creating user interfaces are related to or involve such disciplines as ergonomics and psychology.

A **user interface** is the system by which people (users) interact with a machine. The user interface includes hardware (physical) and software (logical) components. User interfaces exist for various systems, and provide a means of:

- Input, allowing the users to manipulate a system, and/or
- Output, allowing the system to indicate the effects of the users' manipulation.

Generally, the goal of human-machine interaction engineering is to produce a user interface which makes it easy, efficient, and enjoyable to operate a machine in the way which produces the desired result. This generally means that the operator needs to provide minimal input to achieve the desired output, and also that the machine minimizes undesired outputs to the human.

Ever since the increased use of personal computers and the relative decline in societal awareness of heavy machinery, the term user interface has taken on overtones of the (graphical) user interface, while industrial control panel and machinery control design discussions more commonly refer to human-machine interfaces.

Other terms for **user interface** include ***human-computer interface*** (HCI) and ***man-machine interface*** (MMI).

Introduction

To work with a system, users have to be able to control and assess the state of the system. For example, when driving an automobile, the driver uses the steering wheel to control the direction of the vehicle, and the accelerator pedal, brake pedal and gearstick to control the speed of the vehicle. The driver perceives the position of the vehicle by looking through the windshield and exact speed of the vehicle by reading the speedometer. The *user interface of the automobile* is on the whole composed of the instruments the driver can use to accomplish the tasks of driving and maintaining the automobile.

Terminology

There is a distinct difference between **User Interface** versus **Operator Interface** or **Human Machine Interface (HMI)**.

- The term ***user interface*** is often used in the context of (personal) **computer systems** and electronic devices
    - Where a network of equipment or computers are interlinked through an MES (Manufacturing Execution System)-or Host.
    - An HMI is typically local to one machine or piece of equipment, and is the interface method between the human and the equipment/machine. An Operator interface is the interface method by which multiple equipment that are linked by a host control system is accessed or controlled
    - The system may expose several user interfaces to serve different kinds of users. For example, a computerized library database might provide two user interfaces, one for library patrons (limited set of functions, optimized for ease of use) and the other for library personnel (wide set of functions, optimized for efficiency).
- The user interface of a **mechanical system, a vehicle or an industrial installation** is sometimes referred to as the **human-machine interface** (**HMI**). HMI is a modification of the original term MMI (man-machine interface). In practice, the abbreviation MMI is still frequently used although some may claim that MMI stands for something different now. Another abbreviation is HCI, but is more commonly used for human-computer *interaction*. Other terms used are operator interface console (OIC) and operator interface terminal (OIT). However it is abbreviated, the terms refer to the 'layer' that separates a human that is operating a machine from the machine itself.

In science fiction, HMI is sometimes used to refer to what is better described as **direct neural interface**. However, this latter usage is seeing increasing application in the real-life use of (medical) prostheses—the artificial extension that replaces a missing body part (e.g., cochlear implants).

In some circumstance computers might observe the user, and react according to their actions without specific commands. A means of tracking parts of the body is required, and sensors noting the position of the head, direction of gaze and so on have been used experimentally. This is particularly relevant to **immersive interfaces**.

Usability

User interfaces are considered by some authors to be a prime ingredient of Computer user satisfaction.

The design of a user interface affects the amount of effort the user must expend to provide input for the system and to interpret the output of the system, and how much effort it takes to learn how to do this. Usability is the degree to which the design of a particular user interface takes into account the human psychology and physiology of the users, and makes the process of using the system effective, efficient and satisfying.

Usability is mainly a characteristic of the user interface, but is also associated with the functionalities of the product and the process to design it. It describes how well a product can be used for its intended purpose by its target users with efficiency, effectiveness, and satisfaction, also taking into account the requirements from its context of use.

User interfaces in computing

In computer science and human-computer interaction, the *user interface (of a computer program)* refers to the graphical, textual and auditory information the program presents to the user, and the control sequences (such as keystrokes with the computer keyboard, movements of the computer mouse, and selections with the touchscreen) the user employs to control the program.

Types

Currently (as of 2009) the following types of user interface are the most common:

Graphical user interface

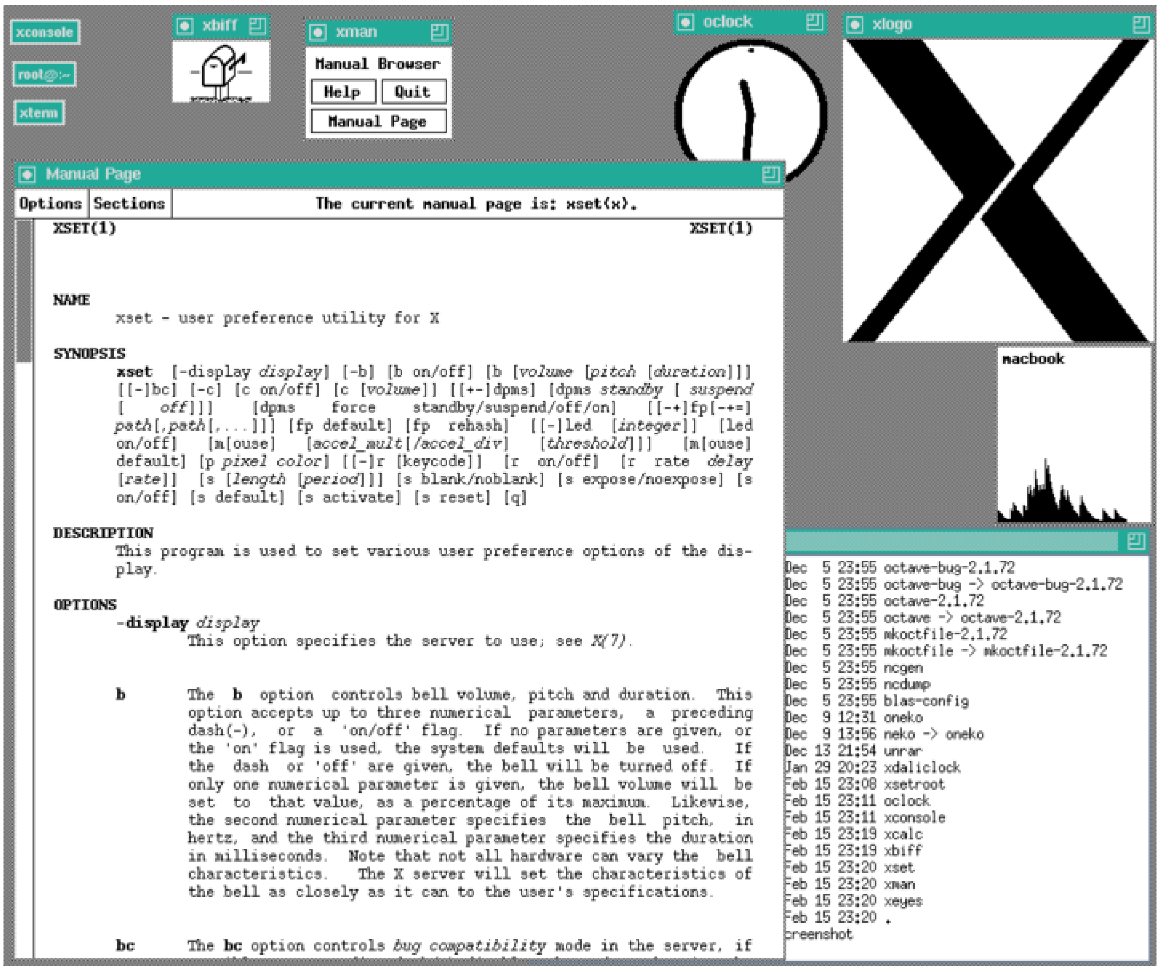
A screenshot of a modern GUI (KDE Plasma Desktop).

The Macintosh 128K was the first commercially successful personal computer to use a graphical user interface

A **graphical user interface** (**GUI**), often pronounced *gooey*, is a type of user interface that allows users to interact with programs in more ways than typing such as computers; hand-held devices such as MP3 players, portable media players or gaming devices; household appliances and office equipment with images rather than text commands. A *GUI* offers graphical icons, and visual indicators, as opposed to text-based interfaces, typed command labels or text navigation to fully represent the information and actions available to a user. The actions are usually performed through direct manipulation of the graphical elements.

The term *GUI* is historically restricted to the scope of two-dimensional display screens with display resolutions capable of describing generic information, in the tradition of the computer science research at the Palo Alto Research Center (PARC). The term *GUI* earlier might have been applicable to other high-resolution types of interfaces that are non-generic, such as videogames, or not restricted to flat screens, like volumetric displays.

History



An early-1990s style Unix desktop running the X Window System graphical user interface

Precursors

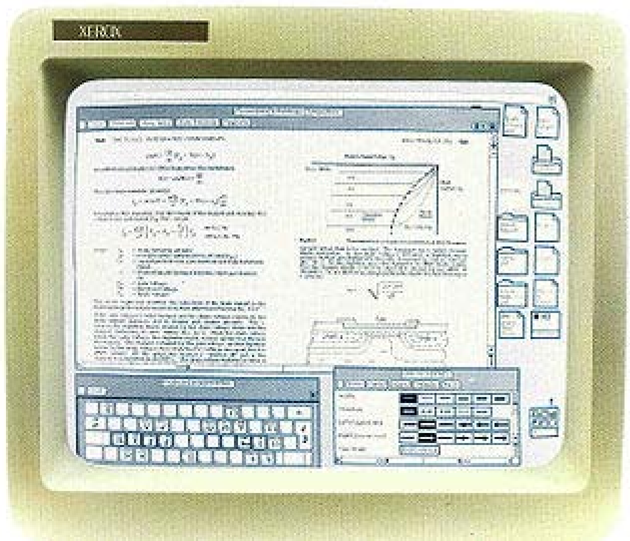
A precursor to GUIs was invented by researchers at the Stanford Research Institute, led by Douglas Engelbart. They developed the use of text-based hyperlinks manipulated with a mouse for the On-Line System. The concept of hyperlinks was further refined and extended to graphics by researchers at Xerox PARC, who went beyond text-based hyperlinks and used a GUI as the primary interface for the Xerox Alto computer. Most modern general-purpose GUIs are derived from this system. As a result, some people call this class of interface a PARC User Interface (PUI) (note that PUI is also an acronym for perceptual user interface).

Ivan Sutherland developed a pointer-based system called the Sketchpad in 1963. It used a light-pen to guide the creation and manipulation of objects in engineering drawings.

PARC user interface

The PARC user interface consisted of graphical elements such as windows, menus, radio buttons, check boxes and icons. The PARC user interface employs a pointing device in addition to a keyboard. These aspects can be emphasized by using the alternative acronym WIMP, which stands for *windows*, *icons*, *menus* and *pointing device*.

Evolution



The Xerox Star Workstation introduced the first GUI operating systems as shown above.

Following PARC the first GUI-centric computer operating model was the Xerox 8010 Star Information System in 1981, followed by the Apple Lisa (which presented the concept of menu bar as well as window controls) in 1983, the Apple Macintosh 128K in 1984, and the Atari ST and Commodore Amiga in 1985.

The GUIs familiar to most people today are Mac OS X, Microsoft Windows, and X Window System interfaces. Apple, IBM and Microsoft used many of Xerox's ideas to develop products, and IBM's Common User Access specifications formed the basis of the user interface found in Microsoft Windows, IBM OS/2 Presentation Manager, and the Unix Motif toolkit and window manager. These ideas evolved to create the interface found in current versions of Microsoft Windows, as well as in Mac OS X and various desktop environments for Unix-like operating systems, such as Linux. Thus most current GUIs have largely common idioms.

Components

A GUI uses a combination of technologies and devices to provide a platform the user can interact with, for the tasks of gathering and producing information.

A series of elements conforming a visual language have evolved to represent information stored in computers. This makes it easier for people with few computer skills to work with and use computer software. The most common combination of such elements in GUIs is the WIMP ("window, icon, menu, pointing device") paradigm, especially in personal computers.

The WIMP style of interaction uses a physical input device to control the position of a cursor and presents information organized in windows and represented with icons. Available commands are compiled together in menus, and actions are performed making gestures with the pointing device. A window manager facilitates the interactions between windows, applications, and the windowing system. The windowing system handles hardware devices such as pointing devices and graphics hardware, as well as the positioning of the cursor.

In personal computers all these elements are modeled through a desktop metaphor, to produce a simulation called a desktop environment in which the display represents a desktop, upon which documents and folders of documents can be placed. Window managers and other software combine to simulate the desktop environment with varying degrees of realism.

Post-WIMP interfaces

Smaller mobile devices such as PDAs and smartphones typically use the WIMP elements with different unifying metaphors, due to constraints in space and available input devices. Applications for which WIMP is not well suited may use newer interaction techniques, collectively named as post-WIMP user interfaces.

Some touch-screen-based operating systems such as Apple's iOS and Android OS currently use post-WIMP styles of interaction. The iPhone's use of more than one finger in contact with the screen allows actions such as pinching and rotating, which are not supported by a single pointer and mouse.

A class of GUIs sometimes referred to as post-WIMP include 3D compositing window manager such as Compiz, Desktop Window Manager, and LG3D. Some post-WIMP interfaces may be better suited for applications which model immersive 3D environments, such as Google Earth.

User interface and interaction design

Designing the visual composition and temporal behavior of GUI is an important part of software application programming. Its goal is to enhance the efficiency and ease of use for the underlying logical design of a stored program, a design discipline known as usability. Techniques of user-centered design are used to ensure that the visual language introduced in the design is well tailored to the tasks it must perform.

Typically, the user interacts with information by manipulating visual widgets that allow for interactions appropriate to the kind of data they hold. The widgets of a well-designed interface are selected to support the actions necessary to achieve the goals of the user. A Model-view-controller allows for a flexible structure in which the interface is independent from and indirectly linked to application functionality, so the GUI can be easily customized. This allows the user to select or design a different *skin* at will, and eases the designer's work to change the interface as the user needs evolve. Nevertheless, good user interface design relates to the user, not the system architecture.

The visible graphical interface features of an application are sometimes referred to as "chrome". Larger widgets, such as windows, usually provide a frame or container for the main presentation content such as a web page, email message or drawing. Smaller ones usually act as a user-input tool.

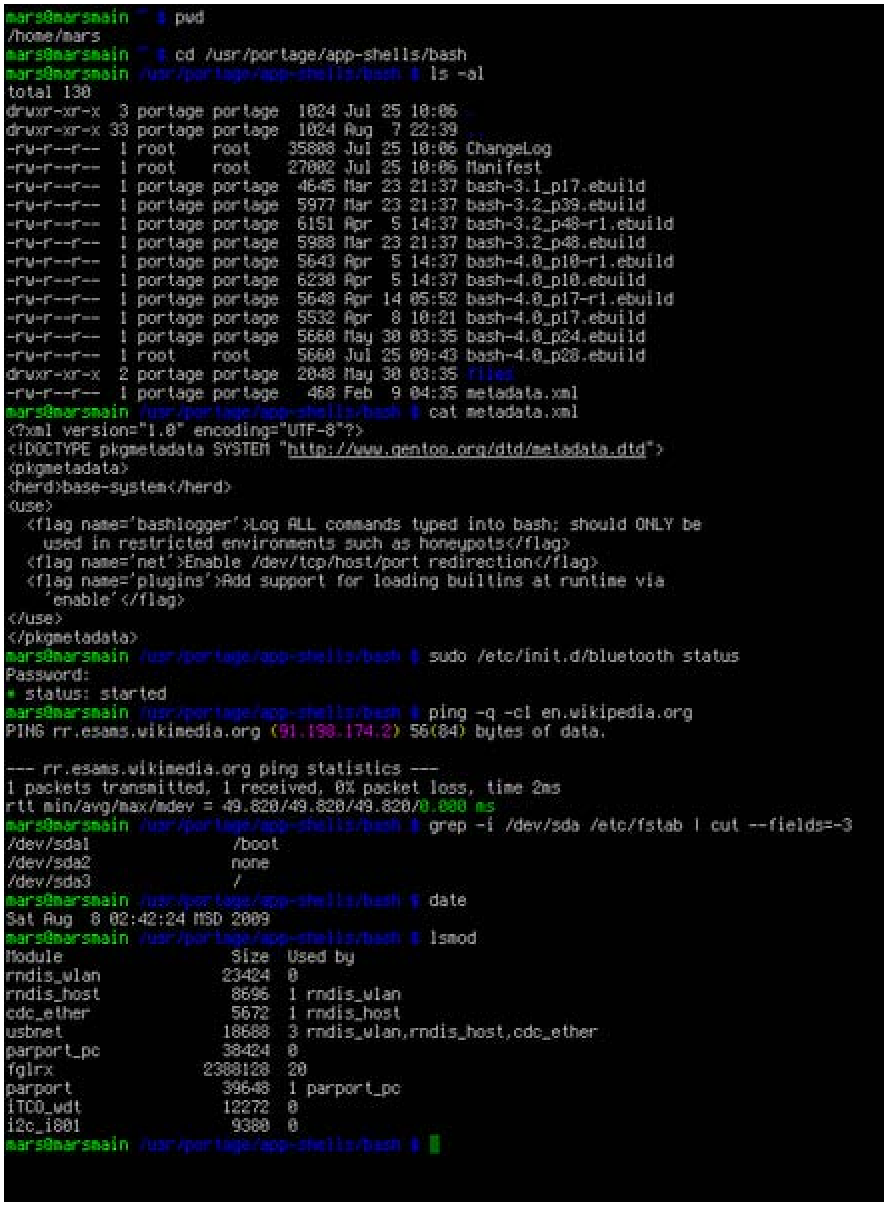
A GUI may be designed for the rigorous requirements of a vertical market. This is known as an "application specific graphical user interface." Among early application specific GUIs was Gene Mosher's 1986 Point of Sale touchscreen GUI. Other examples of an application specific GUIs are:

- Self-service checkouts used in a retail store
- Automated teller machines (ATM)
- Airline self-ticketing and check-in
- Information kiosks in a public space, like a train station or a museum
- Monitors or control screens in an embedded industrial application which employ a real time operating system (RTOS).

The latest cell phones and handheld game systems also employ application specific touchscreen GUIs. Newer automobiles use GUIs in their navigation systems and touch screen multimedia centers.

Comparison to other interfaces

Command-line interfaces



Modern CLI

GUIs were introduced in reaction to the steep learning curve of command-line interfaces (CLI), which require commands to be typed on the keyboard. Since the commands available in command line interfaces can be numerous, complicated operations can be completed using a short sequence of words and symbols. This allows for greater efficiency and productivity once many commands are learned, but reaching this level takes some time because the command words are not easily discoverable and not mnemonic. WIMPs ("window, icon, menu, pointing device"), on the other hand, present the user with numerous widgets that represent and can trigger some of the system's available commands.

WIMPs extensively use modes as the meaning of all keys and clicks on specific positions on the screen are redefined all the time. Command line interfaces use modes only in limited forms, such as the current directory and environment variables.

Most modern operating systems provide both a GUI and some level of a CLI, although the GUIs usually receive more attention. The GUI is usually WIMP-based, although occasionally other metaphors surface, such as those used in Microsoft Bob, 3dwm or File System Visualizer (FSV).

Applications may also provide both interfaces, and when they do the GUI is usually a WIMP wrapper around the command-line version. This is especially common with applications designed for Unix-like operating systems. The latter used to be implemented first because it allowed the developers to focus exclusively on their product's functionality without bothering about interface details such as designing icons and placing buttons. Designing programs this way also allows users to run the program non-interactively, such as in a shell script.

Three-dimensional user interfaces

For typical computer displays, *three-dimensional* is a misnomer—their displays are two-dimensional. Semantically, however, most graphical user interfaces use three dimensions - in addition to height and width, they offer a third dimension of layering or stacking screen elements over one another. This may be represented visually on screen through an illusionary transparent effect, which offers the advantage that information in background windows may still be read, if not interacted with. Or the environment may simply hide the background information, possibly making the distinction apparent by drawing a drop shadow effect over it.

Some environments use the techniques of 3D graphics to project virtual three dimensional user interface objects onto the screen. As the processing power of computer graphics hardware increases, this becomes less of an obstacle to a smooth user experience.

Motivation

Three-dimensional GUIs are quite common in science fiction literature and movies, such as in *Jurassic Park*, which features Silicon Graphics' three-dimensional file manager, "File system navigator", an actual file manager that never got much widespread use as the user interface for a Unix computer. In fiction, three-dimensional user interfaces are often immersible environments like William Gibson's Cyberspace or Neal Stephenson's Metaverse.

Three-dimensional graphics are currently mostly used in computer games, art and computer-aided design (CAD). There have been several attempts at making three-dimensional desktop environments like Sun's Project Looking Glass or SphereXP from Sphere Inc. A three-dimensional computing environment could possibly be used for collaborative work. For example, scientists could study three-dimensional models of molecules in a virtual reality environment, or engineers could work on assembling a three-dimensional model of an airplane. This is a goal of the Croquet project and Project Looking Glass.

Technologies

The use of three-dimensional graphics has become increasingly common in mainstream operating systems, from creating attractive interfaces—eye candy— to functional purposes only possible using three dimensions. For example, user switching is represented by rotating a cube whose faces are each user's workspace, and window management is represented via a Rolodex-style flipping mechanism in Windows Vista. In both cases, the operating system transforms windows on-the-fly while continuing to update the content of those windows.

Interfaces for the X Window System have also implemented advanced three-dimensional user interfaces through compositing window managers such as Beryl, Compiz and KWin using the AIGLX or XGL architectures, allowing for the usage of OpenGL to animate the user's interactions with the desktop.

Another branch in the three-dimensional desktop environment is the three-dimensional GUIs that take the desktop metaphor a step further, like the BumpTop, where a user can manipulate documents and windows as if they were "real world" documents, with realistic movement and physics.

The Zooming User Interface (ZUI) is a related technology that promises to deliver the representation benefits of 3D environments without their usability drawbacks of orientation problems and hidden objects. It is a logical advancement on the GUI, blending some three-dimensional movement with two-dimensional or "2.5D" vector objects.

- **Web-based user interfaces** or **web user interfaces** (WUI) are a subclass of GUIs that accept input and provide output by generating web pages which are transmitted via the Internet and viewed by the user using a web browser program. Newer implementations utilize Java, AJAX, Adobe Flex, Microsoft .NET, or similar technologies to provide real-time control in a separate program, eliminating the need to refresh a traditional HTML based web browser. Administrative web interfaces for web-servers, servers and networked computers are often called Control panels.

- **Touchscreens** are displays that accept input by touch of fingers or a stylus. Used in a growing amount of mobile devices and many types of point of sale, industrial processes and machines, self-service machines etc.

User interfaces that are common in various fields outside desktop computing:

- **Command line interfaces**, where the user provides the input by typing a command string with the computer keyboard and the system provides output by printing text on the computer monitor. Used by programmers and system administrators, in engineering and scientific environments, and by technically advanced personal computer users.

- **Touch user interface** are graphical user interfaces using a touchpad or touchscreen display as a combined input and output device. They supplement or replace other forms of output with haptic feedback methods. Used in computerized simulators etc.

Other types of user interfaces:

- **Attentive user interfaces** manage the user attention deciding when to interrupt the user, the kind of warnings, and the level of detail of the messages presented to the user.
- **Batch interfaces** are non-interactive user interfaces, where the user specifies all the details of the *batch job* in advance to batch processing, and receives the output when all the processing is done. The computer does not prompt for further input after the processing has started.
- **Conversational Interface Agents** attempt to personify the computer interface in the form of an animated person, robot, or other character (such as Microsoft's Clippy the paperclip), and present interactions in a conversational form.
- **Crossing-based interfaces** are graphical user interfaces in which the primary task consists in crossing boundaries instead of pointing.
- **Gesture interface** are graphical user interfaces which accept input in a form of hand gestures, or mouse gestures sketched with a computer mouse or a stylus.
- **Intelligent user interfaces** are human-machine interfaces that aim to improve the efficiency, effectiveness, and naturalness of human-machine interaction by representing, reasoning, and acting on models of the user, domain, task, discourse, and media (e.g., graphics, natural language, gesture).
- **Motion tracking interfaces** monitor the user's body motions and translate them into commands, currently being developed by Apple
- **Multi-screen interfaces**, employ multiple displays to provide a more flexible interaction. This is often employed in computer game interaction in both the commercial arcades and more recently the handheld markets.
- **Noncommand user interfaces**, which observe the user to infer his / her needs and intentions, without requiring that he / she formulate explicit commands.
- **Object-oriented user interface (OOUI)**

- **Reflexive user interfaces** where the users control and redefine the entire system via the user interface alone, for instance to change its command verbs. Typically this is only possible with very rich graphic user interfaces.
- **Tangible user interfaces**, which place a greater emphasis on touch and physical environment or its element.
- **Task-Focused Interfaces** are user interfaces which address the information overload problem of the desktop metaphor by making tasks, not files, the primary unit of interaction
- **Text user interfaces** are user interfaces which output text, but accept other form of input in addition to or in place of typed command strings.
- **Voice user interfaces**, which accept input and provide output by generating voice prompts. The user input is made by pressing keys or buttons, or responding verbally to the interface.
- **Natural-Language interfaces** - Used for search engines and on webpages. User types in a question and waits for a response.
- **Zero-Input interfaces** get inputs from a set of sensors instead of querying the user with input dialogs.
- **Zooming user interfaces** are graphical user interfaces in which information objects are represented at different levels of scale and detail, and where the user can change the scale of the viewed area in order to show more detail.

Consistency

A key property of a good user interface is consistency. Good user interface design is about getting a user to have a consistent set of expectations, and then meeting those expectations.

There are three important aspects to consistency.

First, the controls for different features should be presented in a consistent manner so that users can find the controls easily. For example, users find it difficult to use software when some commands are available through menus, some through icons, some through right-clicks, some under a separate button at one corner of a screen, some grouped by function, some grouped by “common,” some grouped by “advanced.” A user looking for a command should have a consistent search strategy for finding it. The more search strategies a user has to use, the more frustrating the search will be. The more consistent the grouping, the easier the search.

Second, the "principle of least astonishment" is crucial. Various features should work in similar ways. For example, some features in Adobe Acrobat are "select tool, then select text to which apply." Others are "select text, then apply action to selection." . Commands should work the same way in all contexts.

Third, consistency counsels against user interface changes version-to-version. Change should be minimized, and forward-compatibility should be maintained. For example, the change from the menu bars of Microsoft Office 2003 to the ribbon toolbar of Microsoft

Office 2007 caused mixed reactions to the redesign, intended to enhance access to the most used functions. It was said to cause "anger and frustration," and "major efforts in time, training and costs." Power users said it "takes too much time and patience to learn" the new interface. An online survey by an Excel user group reports that about 80% of respondents had a negative opinion of the change, and within that 80%, the self-estimated reduction in productivity was "about 35%".

Consistency is one quality to trade off in user interface design—not the only thing, but one of the most important. In some cases, a violation of consistency principles can provide sufficiently clear advantages that a wise and careful user interface designer may choose to violate consistency to achieve some other important goal. Generally, less mature software has fewer users who are entrenched in the status quo. Older, more broadly used software must more carefully hew to the status quo to avoid disruptive costs. The most experienced users, and the ones who derive most value from a program, are the users who tend to bear the greatest costs from change. However, of those trade-offs, consistency is one of the most important core principles, and it should be violated least often. Bad user interface design, and poorly implemented changes to an existing user interface, can impose staggering costs on users.

Modalities and modes

A **modality** is a path of communication employed by the user interface to carry input and output. Examples of modalities:

- Input — computer keyboard allows the user to enter typed text, digitizing tablet allows the user to create free-form drawing
- Output — computer monitor allows the system to display text and graphics (*vision modality*), loudspeaker allows the system to produce sound (*auditory modality*)

The user interface may employ several redundant input modalities and output modalities, allowing the user to choose which ones to use for interaction.

A **mode** is a distinct method of operation within a computer program, in which the same input can produce different perceived results depending of the state of the computer program. For example, caps lock sets an input mode in which typed letters are uppercase by default; the same typing produces lowercase letters when not in caps lock mode. Heavy use of modes often reduces the usability of a user interface, as the user must expend effort to remember current mode states, and switch between mode states as necessary.

3D Interaction

In computing, **3D interaction** is a form of Human-machine interaction where users are able to move and perform interaction in 3D space. Both human and machine process information where the physical position of elements in the 3D space is relevant.

The 3D space used for interaction can be the real physical space, a virtual space representation simulated in the computer, or a combination of both. When the real space is used for data input, humans perform actions or give commands to the machine using an input device that detects the 3D position of the human action. When it's used for data output, the simulated 3D virtual scene is projected onto the real environment through one output device or a combination of them.

Background

3D's early beginnings can be traced back to 1962 when Morton Heilig invented the *Sensorama* simulator. It provided 3D video feedback, as well motion, audio, and haptic feedbacks to produce a virtual environment. The next stage of development was Dr. Ivan Sutherland's completion of his pioneering work in 1968. He created a head-mounted display that produced a 3D, virtual environment by presenting a left and right still image of that environment.

Availability of technology as well as impractical costs held back the development and application of virtual environments until the 1980s. Applications were limited to military ventures in the United States. Since then, further research and technological advancements have allowed new doors to be opened to application in various other areas such as education, entertainment, and manufacturing.

3D interaction

In *3D Interaction*, users carry out their tasks and perform functions by exchanging information with computer systems in 3D space. It is an intuitive type of interaction because humans interact in three dimensions in the real world. The tasks that users perform have been classified as selection and manipulation of objects in virtual space, navigation, and system control. Tasks can be performed in virtual space through interaction techniques and by utilizing interaction devices. 3D interaction techniques were classified according to the task group it supports. Techniques that support navigation tasks are classified as *navigation techniques*. Techniques that support object selection and manipulation are labeled *selection and manipulation techniques*. Lastly, *system control techniques* support tasks that have to do with controlling the application itself. A consistent and efficient mapping between techniques and interaction devices must be made in order for the system to be usable and effective. Interfaces associated with 3D interaction are called *3D interfaces*. Like other types of user interfaces, it involves two-way communication between users and system, but allows users to perform action in 3D space. Input devices permit the users to give directions and commands to the system, while output devices allow the machine to present information back to them.

3D Interfaces have been used in applications that feature virtual environments, and augmented and mixed realities. In virtual environments, users may interact directly with the environment or use tools with specific functionalities to do so. 3D interaction occurs when physical tools are controlled in 3D spatial context to control a corresponding virtual tool.

Users experience a sense of presence when engaged in an immersive virtual world. Enabling the users to interact with this world in 3D allows them to make use of natural and intrinsic knowledge of how information exchange takes place with physical objects in the real world. Texture, sound, and speech can all be used to augment 3D interaction. Currently, users still have difficulty in interpreting 3D space visuals and understanding how interaction occurs. Although it's a natural way for humans to move around in a three-dimensional world, the difficulty exists because many of the cues present in real environments are missing from virtual environments. Perception and occlusion are the primary perceptual cues used by humans. Also, even though scenes in virtual space appear three-dimensional, they are still displayed on a 2D surface so some inconsistencies in depth perception will still exist.

3D user interfaces

User interfaces are the means for communication between users and systems. 3D interfaces include media for 3D representation of system state, and media for 3D user input or manipulation. Using 3D representations is not enough to create 3D interaction. The users must have a way of performing actions in 3D as well. To that effect, special input and output devices have been developed to support this type of interaction. Some, such as the 3D mouse, were developed based on existing devices for 2D interaction.

Input Devices

Input Devices are instruments used to manipulate objects, and send control instructions to the computer system. They vary in terms of degrees of freedom available to them and can be classified into standard input devices, trackers, control devices, navigation equipment, and gesture interfaces.

Standard input devices include keyboards, tablets and stylus, joysticks, mice, touch screens, knobs, and trackballs.

Trackers detect or monitor head, hand or body movements and send that information to the computer. The computer then translates it and ensures that position and orientation are reflected accurately in the virtual world. Tracking is important in presenting the correct viewpoint, coordinating the spatial and sound information presented to users as well the tasks or functions that they could perform. 3D trackers have been identified as mechanical, magnetic, ultrasonic, optical, and hybrid inertial. Examples of trackers include motion trackers, eye trackers, and data gloves.

A simple 2D mouse may be considered a navigation device if it allows the user to move to a different location in a virtual 3D space. Navigation devices such as the treadmill and bicycle make use of the natural ways that humans travel in the real world. Treadmills simulate walking or running and bicycles or similar type equipment simulate vehicular travel. In the case of navigation devices, the information passed on to the machine is the user's location and movements in virtual space.

Wired gloves and bodysuits allow gestural interaction to occur. These send hand or body position and movement information to the computer using sensors.

Output Devices

Output devices allow the machine to provide information or feedback to the user. They include visual displays, auditory displays, and haptic displays. Visual displays provide feedback to users in 3D visual form. Head-mounted displays and CAVEs (Cave Automatic Virtual Environment) are examples of a fully-immersive visual display, where the user can see only the virtual world and not the real world. Semi-immersive displays allow users to see both. Monitors and workbenches are examples of semi-immersive displays. Auditory displays provide information in auditory form. This is especially useful when supplying location and spatial information to the users. Adding background audio component to a display adds to the sense of realism. Haptic displays send tactile feedback or feeling back to the user.

3D Interaction Techniques

3D interaction techniques are methods used in order to execute different types of task in 3D space. Techniques are classified according to the tasks that they support.

Selection and Manipulation

Users need to be able to manipulate virtual objects. Manipulation tasks involve selecting and moving an object. Sometimes, rotation of the object is involved as well. Direct-hand manipulation is the most natural technique because manipulating physical objects with the hand is intuitive for humans. However, this is not always possible. A virtual hand that can select and re-locate virtual objects will work as well. 3D widgets can be used to put controls on objects. Users can employ these to re-locate or re-orient an object. Other techniques include the Go-Go technique and ray casting, where a virtual ray is used to point to, and select and object. More recently there has been user interface development and research by Richard White in Kansas over the past 3 years regarding interactive surfaces & classroom interactive whiteboards, grade school students, and 3D natural user interfaces known as Edusim.

Navigation

The computer needs to provide the user with information regarding location and movement. Navigation tasks have two components. Travel involves moving from the current location to the desired point. Wayfinding refers to finding and setting routes to get to a travel goal within the virtual environment.

- **Wayfinding** : Wayfinding in virtual space is different and more difficult to do than in the real world because synthetic environments are often missing perceptual cues and movement constraints. It can be supported using user-centred techniques such as using a larger field of view and supplying motion cues, or environment-centred techniques like structural organization and wayfinding principles.

- **Travel** : Good travel techniques allow the user to easily move through the environment. There are three types of travel tasks namely, exploration, search, and manoeuvring. Travel techniques can be classified into the following five categories:
    - Physical movement – user moves through the virtual world
    - Manual Viewpoint manipulation – use hand motions to achieve movement
    - Steering – direction specification
    - Target-based travel – destination specification
    - Route planning – path specification

System control

Tasks that involve issuing commands to the application in order to change system mode or activate some functionality fall under the category of system control. Techniques that support system control tasks in three-dimensions are classified as:

- Graphical menus
- Voice commands
- Gestural interaction
- Virtual tools with specific functions

Symbolic Input

This task allows the user to enter and/or edit, for example, text, making it possible to annotate 3D scenes or 3D objects.

Chapter 4

Brain–Computer Interface and End-User Computing

Brain–computer interface

A **brain–computer interface** (**BCI**), sometimes called a **direct neural interface** or a **brain–machine interface**, is a direct communication pathway between a brain and an external device. BCIs are often aimed at assisting, augmenting or repairing human cognitive or sensory-motor functions.

Research on BCIs began in the 1970s at the University of California Los Angeles (UCLA) under a grant from the National Science Foundation, followed by a contract from DARPA. The papers published after this research also mark the first appearance of the expression *brain–computer interface* in scientific literature.

The field of BCI has since advanced mostly toward neuroprosthetics applications that aim at restoring damaged hearing, sight and movement. Thanks to the remarkable cortical plasticity of the brain, signals from implanted prostheses can, after adaptation, be handled by the brain like natural sensor or effector channels. Following years of animal experimentation, the first neuroprosthetic devices implanted in humans appeared in the mid-nineties.

BCI versus neuroprosthetics

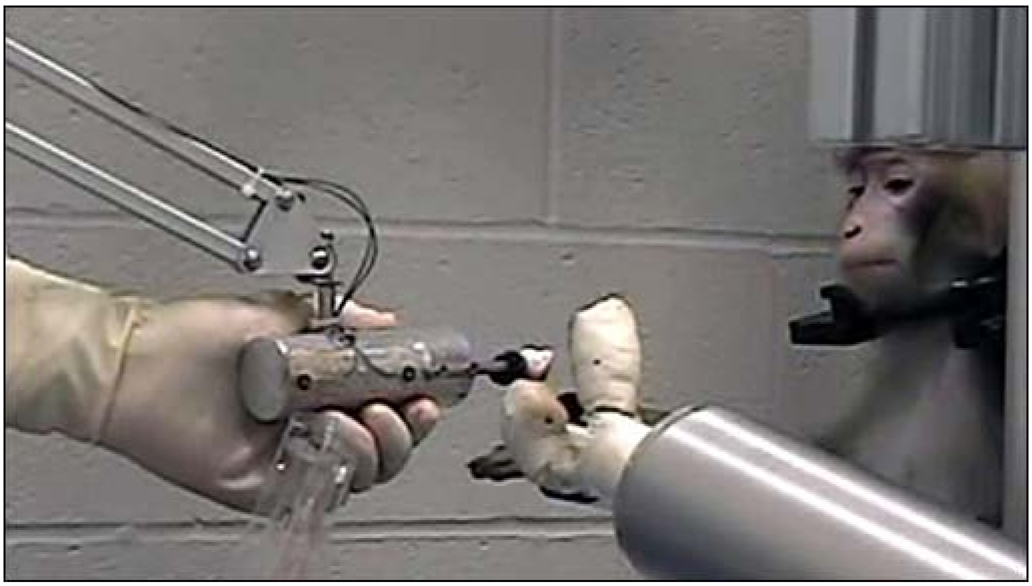
Neuroprosthetics is an area of neuroscience concerned with neural prostheses—using artificial devices to replace the function of impaired nervous systems or sensory organs. The most widely used neuroprosthetic device is the cochlear implant, which, as of 2006, has been implanted in approximately 100,000 people worldwide. There are also several neuroprosthetic devices that aim to restore vision, including retinal implants.

The differences between BCIs and neuroprosthetics are mostly in the ways the terms are used: neuroprosthetics typically connect the nervous system to a device, whereas BCIs usually connect the brain (or nervous system) with a computer system. Practical neuroprosthetics can be linked to any part of the nervous system—for example, peripheral nerves—while the term "BCI" usually designates a narrower class of systems which interface with the central nervous system.

The terms are sometimes used interchangeably, and for good reason. Neuroprosthetics and BCIs seek to achieve the same aims, such as restoring sight, hearing, movement, ability to communicate, and even cognitive function. Both use similar experimental methods and surgical techniques.

.

Early work



Monkey operating a robotic arm with brain–computer interfacing

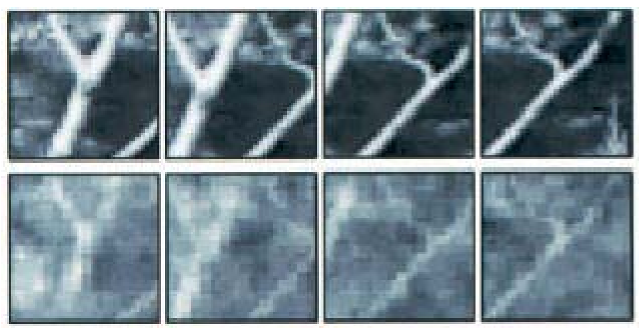
The operant conditioning studies of Fetz and colleagues first showed that monkeys could learn to control the deflection of a biofeedback meter arm with neural activity . Such work in the 1970s established that monkeys could quickly learn to voluntarily control the firing rates of individual and multiple neurons in the primary motor cortex if they were rewarded for generating appropriate patterns of neural activity.

Studies that developed algorithms to reconstruct movements from motor cortex neurons, which control movement, date back to the 1970s. In the 1980s, Apostolos Georgopoulos at Johns Hopkins University found a mathematical relationship between the electrical responses of single motor-cortex neurons in rhesus macaque monkeys and the direction that monkeys moved their arms (based on a cosine function). He also found that dispersed groups of neurons in different areas of the brain collectively controlled motor commands but was only able to record the firings of neurons in one area at a time because of technical limitations imposed by his equipment.

There has been rapid development in BCIs since the mid-1990s. Several groups have been able to capture complex brain motor centre signals using recordings from neural ensembles (groups of neurons) and use these to control external devices, including research groups led by Richard Andersen, John Donoghue, Phillip Kennedy, Miguel Nicolelis, and Andrew Schwartz.

Prominent research successes

Phillip Kennedy and colleagues built the first intracortical brain–computer interface by implanting neurotrophic-cone electrodes into monkeys.



Yang Dan and colleagues' recordings of cat vision using a BCI implanted in the lateral geniculate nucleus (top row: original image; bottom row: recording)

In 1999, researchers led by Yang Dan at University of California, Berkeley decoded neuronal firings to reproduce images seen by cats. The team used an array of electrodes embedded in the thalamus (which integrates all of the brain's sensory input) of sharp-eyed cats. Researchers targeted 177 brain cells in the thalamus lateral geniculate nucleus area, which decodes signals from the retina. The cats were shown eight short movies, and their neuron firings were recorded. Using mathematical filters, the researchers decoded the signals to generate movies of what the cats saw and were able to reconstruct recognizable scenes and moving objects. Similar results in humans have been since then achieved by researchers in Japan (see below).

Miguel Nicolelis has been a prominent proponent of using multiple electrodes spread over a greater area of the brain to obtain neuronal signals to drive a BCI. Such neural ensembles are said to reduce the variability in output produced by single electrodes, which could make it difficult to operate a BCI.

After conducting initial studies in rats during the 1990s, Nicolelis and his colleagues developed BCIs that decoded brain activity in owl monkeys and used the devices to reproduce monkey movements in robotic arms. Monkeys have advanced reaching and grasping abilities and good hand manipulation skills, making them ideal test subjects for this kind of work.

By 2000, the group succeeded in building a BCI that reproduced owl monkey movements while the monkey operated a joystick or reached for food. The BCI operated in real time and could also control a separate robot remotely over Internet protocol. But the monkeys could not see the arm moving and did not receive any feedback, a so-called open-loop BCI.

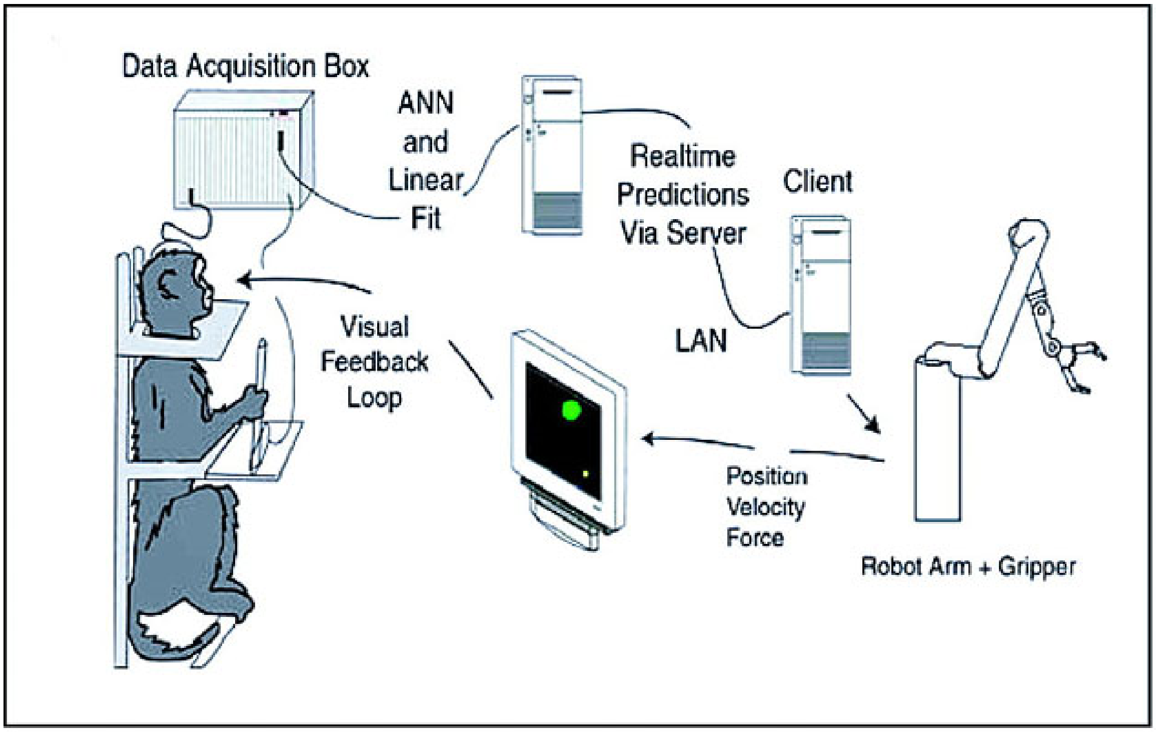


Diagram of the BCI developed by Miguel Nicolelis and colleagues for use on Rhesus monkeys

Later experiments by Nicolelis using rhesus monkeys, succeeded in closing the feedback loop and reproduced monkey reaching and grasping movements in a robot arm. With their deeply cleft and furrowed brains, rhesus monkeys are considered to be better models for human neurophysiology than owl monkeys. The monkeys were trained to reach and grasp objects on a computer screen by manipulating a joystick while corresponding movements by a robot arm were hidden. The monkeys were later shown the robot directly and learned to control it by viewing its movements. The BCI used velocity predictions to control reaching movements and simultaneously predicted hand gripping force.

Other labs that develop BCIs and algorithms that decode neuron signals include John Donoghue from Brown University, Andrew Schwartz from the University of Pittsburgh and Richard Andersen from Caltech. These researchers were able to produce working BCIs even though they recorded signals from far fewer neurons than Nicolelis (15–30 neurons versus 50–200 neurons).

Donoghue's group reported training rhesus monkeys to use a BCI to track visual targets on a computer screen with or without assistance of a joystick (closed-loop BCI). Schwartz's group created a BCI for three-dimensional tracking in virtual reality and also reproduced BCI control in a robotic arm. The group created headlines when they demonstrated that a monkey could feed itself pieces of zucchini using a robotic arm controlled by the animal's own brain signals.

Andersen's group used recordings of premovement activity from the posterior parietal cortex in their BCI, including signals created when experimental animals anticipated receiving a reward.

In addition to predicting kinematic and kinetic parameters of limb movements, BCIs that predict electromyographic or electrical activity of muscles are being developed. Such BCIs could be used to restore mobility in paralyzed limbs by electrically stimulating muscles.

Miguel Nicolelis *et al.* showed that activity of large neural ensembles can predict arm position. This work made possible creation of brain–machine interfaces — electronic devices that read arm movement intentions and translate them into movements of artificial actuators. Carmena et al. programmed the neural coding in a brain–machine interface allowed a monkey to control reaching and grasping movements by a robotic arm, and Lebedev *et al.* argued that brain networks reorganize to create a new representation of the robotic appendage in addition to the representation of the animal's own limbs.

The biggest impediment of BCI technology at present is the lack of a sensor modality that provides safe, accurate, and robust access to brain signals. It is conceivable or even likely that such a sensor will be developed within the next twenty years. The use of such a sensor should greatly expand the range of communication functions that can be provided using a BCI.

Development and implementation of a Brain–Computer Interface (BCI) system is complex and time consuming. In response to this problem, Dr. Gerwin Schalk has been developing a general-purpose system for BCI research, called BCI2000. BCI2000 has been in development since 2000 in a project led by the Brain–Computer Interface R&D Program at the Wadsworth Center of the New York State Department of Health in Albany, New York, USA.

A new 'wireless' approach uses light-gated ion channels such as Channelrhodopsin to control the activity of genetically defined subsets of neurons in vivo. In the context of a simple learning task, illumination of transfected cells in the somatosensory cortex influenced the decision making process of freely moving mice.

Human BCI research

Invasive BCIs

Invasive BCI research has targeted repairing damaged sight and providing new functionality to persons with paralysis. Invasive BCIs are implanted directly into the grey matter of the brain during neurosurgery. As they rest in the grey matter, invasive devices produce the highest quality signals of BCI devices but are prone to scar-tissue build-up, causing the signal to become weaker or even lost as the body reacts to a foreign object in the brain.

Jens Naumann, a man with acquired blindness, being interviewed about his vision BCI on CBS's The Early Show

In *vision science*, direct brain implants have been used to treat non-congenital (acquired) blindness. One of the first scientists to come up with a working brain interface to restore sight was private researcher William Dobelle.

Dobelle's first prototype was implanted into "Jerry", a man blinded in adulthood, in 1978. A single-array BCI containing 68 electrodes was implanted onto Jerry’s visual cortex and succeeded in producing phosphenes, the sensation of seeing light. The system included cameras mounted on glasses to send signals to the implant. Initially, the implant allowed Jerry to see shades of grey in a limited field of vision at a low frame-rate. This also required him to be hooked up to a two-ton mainframe, but shrinking electronics and faster computers made his artificial eye more portable and now enable him to perform simple tasks unassisted.

Dummy unit illustrating the design of a BrainGate interface

In 2002, Jens Naumann, also blinded in adulthood, became the first in a series of 16 paying patients to receive Dobelle's second generation implant, marking one of the earliest commercial uses of BCIs. The second generation device used a more sophisticated implant enabling better mapping of phosphenes into coherent vision. Phosphenes are spread out across the visual field in what researchers call the starry-night effect. Immediately after his implant, Jens was able to use his imperfectly restored vision to drive slowly around the parking area of the research institute.

BCIs focusing on *motor neuroprosthetics* aim to either restore movement in individuals with paralysis or provide devices to assist them, such as interfaces with computers or robot arms.

Researchers at Emory University in Atlanta led by Philip Kennedy and Roy Bakay were first to install a brain implant in a human that produced signals of high enough quality to simulate movement. Their patient, Johnny Ray (1944–2002), suffered from 'locked-in syndrome' after suffering a brain-stem stroke in 1997. Ray's implant was installed in 1998 and he lived long enough to start working with the implant, eventually learning to control a computer cursor; he died in 2002 of a brain aneurysm.

Tetraplegic Matt Nagle became the first person to control an artificial hand using a BCI in 2005 as part of the first nine-month human trial of Cyberkinetics Neurotechnology's BrainGate chip-implant. Implanted in Nagle's right precentral gyrus (area of the motor cortex for arm movement), the 96-electrode BrainGate implant allowed Nagle to control a robotic arm by thinking about moving his hand as well as a computer cursor, lights and TV. One year later, professor Jonathan Wolpaw received the prize of the Altran Foundation for Innovation to develop a Brain Computer Interface with electrodes located on the surface of the skull, instead of directly in the brain.

Partially-invasive BCIs

Partially invasive BCI devices are implanted inside the skull but rest outside the brain rather than within the grey matter. They produce better resolution signals than non-invasive BCIs where the bone tissue of the cranium deflects and deforms signals and have a lower risk of forming scar-tissue in the brain than fully-invasive BCIs.

Electrocorticography (ECoG) measures the electrical activity of the brain taken from beneath the skull in a similar way to non-invasive electroencephalography (see below), but the electrodes are embedded in a thin plastic pad that is placed above the cortex, beneath the dura mater. ECoG technologies were first trialed in humans in 2004 by Eric Leuthardt and Daniel Moran from Washington University in St Louis. In a later trial, the researchers enabled a teenage boy to play Space Invaders using his ECoG implant. This research indicates that control is rapid, requires minimal training, and may be an ideal tradeoff with regards to signal fidelity and level of invasiveness.

(Note: These electrodes were not implanted in the patients for BCI experiments. The patient was suffering from severe epilepsy and had the electrodes temporarily implanted to help his physicians localize seizure foci; the researchers simply took advantage of this.)

Light Reactive Imaging BCI devices are still in the realm of theory. These would involve implanting a laser inside the skull. The laser would be trained on a single neuron and the neuron's reflectance measured by a separate sensor. When the neuron fires, the laser light pattern and wavelengths it reflects would change slightly. This would allow researchers to monitor single neurons but require less contact with tissue and reduce the risk of scar-tissue build-up.

This signal can be either subdural or epidural, but is not taken from within the brain parenchyma itself. It has not been studied extensively until recently due to the limited access of subjects. Currently, the only manner to acquire the signal for study is through the use of patients requiring invasive monitoring for localization and resection of an epileptogenic focus.

ECoG is a very promising intermediate BCI modality because it has higher spatial resolution, better signal-to-noise ratio, wider frequency range, and lesser training requirements than scalp-recorded EEG, and at the same time has lower technical difficulty, lower clinical risk, and probably superior long-term stability than intracortical single-neuron recording. This feature profile and recent evidence of the high level of control with minimal training requirements shows potential for real world application for people with motor disabilities.

Non-invasive BCIs

As well as invasive experiments, there have also been experiments in humans using non-invasive neuroimaging technologies as interfaces. Signals recorded in this way have been used to power muscle implants and restore partial movement in an experimental volunteer. Although they are easy to wear, non-invasive implants produce poor signal resolution because the skull dampens signals, dispersing and blurring the electromagnetic waves created by the neurons. Although the waves can still be detected it is more difficult to determine the area of the brain that created them or the actions of individual neurons.

EEG

Recordings of brainwaves produced by an electroencephalogram

Electroencephalography (EEG) is the most studied potential non-invasive interface, mainly due to its fine temporal resolution, ease of use, portability and low set-up cost. But as well as the technology's susceptibility to noise, another substantial barrier to using EEG as a brain–computer interface is the extensive training required before users can work the technology. For example, in experiments beginning in the mid-1990s, Niels Birbaumer of the University of Tübingen in Germany trained severely paralysed people to self-regulate the *slow cortical potentials* in their EEG to such an extent that these signals could be used as a binary signal to control a computer cursor. (Birbaumer had earlier trained epileptics to prevent impending fits by controlling this low voltage wave.) The experiment saw ten patients trained to move a computer cursor by controlling their brainwaves. The process was slow, requiring more than an hour for patients to write 100 characters with the cursor, while training often took many months.

Another research parameter is the type of waves measured. Birbaumer's later research with Jonathan Wolpaw at New York State University has focused on developing technology that would allow users to choose the brain signals they found easiest to operate a BCI, including *mu* and *beta* rhythms.

A further parameter is the method of feedback used and this is shown in studies of P300 signals. Patterns of P300 waves are generated involuntarily (stimulus-feedback) when people see something they recognize and may allow BCIs to decode categories of thoughts without training patients first. By contrast, the biofeedback methods described above require learning to control brainwaves so the resulting brain activity can be detected.

Lawrence Farwell and Emanuel Donchin developed an EEG-based brain–computer interface in the 1980s. Their "mental prosthesis" used the P300 brainwave response to allow subjects, including one paralyzed Locked-In syndrome patient, to communicate words, letters, and simple commands to a computer and thereby to speak through a speech synthesizer driven by the computer. A number of similar devices have been developed since then. In 2000, for example, research by Jessica Bayliss at the University of Rochester showed that volunteers wearing virtual reality helmets could control elements in a virtual world using their P300 EEG readings, including turning lights on and off and bringing a mock-up car to a stop.

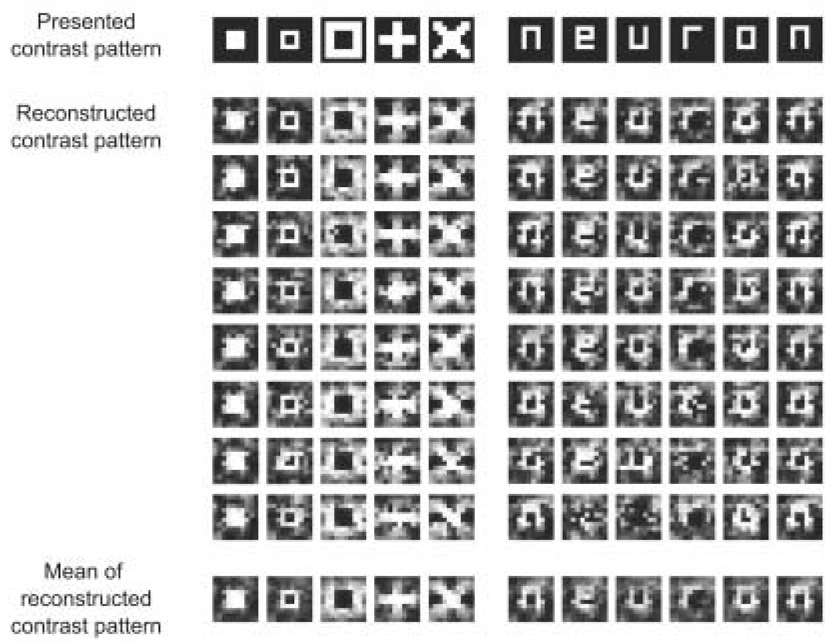
In the early 90s Babak Taheri, at UC DAVIS demonstrated the first single and also multichannel dry active electrode arrays using micro-machining. The single channel dry EEG electrode construction and results were published in 1994. The arrayed electrode was also demonstrated to perform well compared to Ag/AgCl electrodes. The device consisted of four sites of sensors with integrated electronics to reduce noise by impedance matching. The advantages of such electrodes are: (1) no electrolyte used, (2) no skin preparation, (3) significantly reduced sensor size, and (4) compatibility with EEG monitoring systems. The active electrode array is an integrated system made of an array of capacitive sensors with local integrated circuitry housed in a package with batteries to power the circuitry. This level of integration was required to achieve the functional performance obtained by the electrode. The electrode was tested on an electrical test bench and on human subjects in four modalities of EEG activity, namely: (1) spontaneous EEG, (2) sensory event-related potentials, (3) brain stem potentials, and (4) cognitive event-related potentials. The performance of the dry electrode compared favorably with that of the standard wet Ag/AgCl electrodes in terms of skin preparation, no gel requirements (dry), and higher signal-to-noise ratio. In 1999, researchers at Case Western Reserve University led by Hunter Peckham, used 64-electrode EEG skullcap to return limited hand movements to quadriplegic Jim Jatich. As Jatich concentrated on simple but opposite concepts like up and down, his beta-rhythm EEG output was analysed using software to identify patterns in the noise. A basic pattern was identified and used to control a switch: Above average activity was set to on, below average off. As well as enabling Jatich to control a computer cursor the signals were also used to drive the nerve controllers embedded in his hands, restoring some movement.

Electronic neural networks have been deployed which shift the learning phase from the user to the computer. Experiments by scientists at the Fraunhofer Society in 2004 using neural networks led to noticeable improvements within 30 minutes of training.

Experiments by Eduardo Miranda aim to use EEG recordings of mental activity associated with music to allow the disabled to express themselves musically through an encephalophone.

The Emotiv company has been selling a commercial video game controller, known as the Epoc, since December 2009. The Epoc uses electromagnetic sensors.

MEG and MRI



ATR Labs' reconstruction of human vision using fMRI (top row: original image; bottom row: reconstruction from mean of combined readings)

Magnetoencephalography (MEG) and functional magnetic resonance imaging (fMRI) have both been used successfully as non-invasive BCIs. In a widely reported experiment, fMRI allowed two users being scanned to play Pong in real-time by altering their haemodynamic response or brain blood flow through biofeedback techniques.

fMRI measurements of haemodynamic responses in real time have also been used to control robot arms with a seven second delay between thought and movement.

More recently, research developed in the Advanced Telecommunications Research (ATR) Computational Neuroscience Laboratories in Kyoto, Japan allowed the scientists to reconstruct images directly from the brain and display them on a computer. The article announcing these achievements was the cover story of the journal Neuron of 10 December 2008, While the early results are limited to black and white images of 10x10 squares (pixels), according to the researchers further development of the technology may make it possible to achieve color images, and even view or record dreams.

Commercialization and companies

John Donoghue and fellow researchers founded Cyberkinetics. The company markets its electrode arrays under the BrainGate product name and has set the development of practical BCIs for humans as its major goal. The BrainGate is based on the Utah Array developed by Dick Normann.

Philip Kennedy founded Neural Signals in 1987 to develop BCIs that would allow paralysed patients to communicate with the outside world and control external devices. As well as an invasive BCI, the company also sells an implant to restore speech. Neural Signals' Brain Communicator BCI device uses glass cones containing microelectrodes coated with proteins to encourage the electrodes to bind to neurons.

Although 16 paying patients were treated using William Dobelle's vision BCI, new implants ceased within a year of Dobelle's death in 2004. A company controlled by Dobelle, Avery Biomedical Devices, and Stony Brook University are continuing development of the implant, which has not yet received Food and Drug Administration approval in the United States for human implantation.

Ambient, at a TI developers conference in early 2008, demoed a product they have in development call The Audeo. The Audeo is being developed to create a human–computer interface for communication without the need of physical motor control or speech production. Using signal processing, unpronounced speech representing the thought of the mind can be translated from intercepted neurological signals.

Mindball is a product developed and commercialized by Interactive Productline in which players compete to control a ball's movement across a table by becoming more relaxed and focused. Interactive Productline is a Swedish company whose objective is to develop and sell easy understandable EEG products that train the ability to relax and focus.

An Austrian company, Guger Technologies, g.tec, has been offering Brain Computer Interface systems since 1999. The company provides base BCI models as development platforms for the research community to build upon, including the P300 Speller, Motor Imagery, and mu-rhythm. They commercialized a Steady State Visual Evoked Potiential BCI solution in 2008 with 4 degrees of machine control.

A Spanish company, Starlab, has entered this market in 2009 with a wireless 4-channel system called ENOBIO. Designed for research purposes the system provides a platform for application development.

There are three main consumer-devices commercial-competitors in this area (launch date mentioned in brackets) which have launched such devices primarily for gaming- and PC-users:

- Neural Impulse Actuator (April, 2008)
- Emotiv Systems (December, 2009)

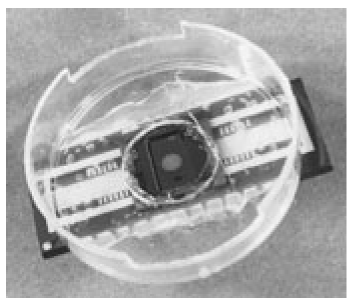
- NeuroSky (MindSet – June, 2009; Uncle Milton Force Trainer – Fall 2009, Mattel MindFlex – Summer, 2009)

Synthetic telepathy

Research is ongoing into synthetic or computer-mediated telepathy which would allow user-to-user communication through analysis of neural signals. The research aims to detect and analyze the word-specific neural signals, using EEG, which occur before speech is vocalized, and to see if the patterns are generalizable. As of 2009, the research is focused on military uses.

Cell-culture BCIs

Researchers have built devices to interface with neural cells and entire neural networks in cultures outside animals. As well as furthering research on animal implantable devices, experiments on cultured neural tissue have focused on building problem-solving networks, constructing basic computers and manipulating robotic devices. Research into techniques for stimulating and recording from individual neurons grown on semiconductor chips is sometimes referred to as neuroelectronics or neurochips.



World first: Neurochip developed by Caltech researchers Jerome Pine and Michael Maher

Development of the first working neurochip was claimed by a Caltech team led by Jerome Pine and Michael Maher in 1997. The Caltech chip had room for 16 neurons.

In 2003, a team led by Theodore Berger at the University of Southern California started work on a neurochip designed to function as an artificial or prosthetic hippocampus. The neurochip was designed to function in rat brains and is intended as a prototype for the eventual development of higher-brain prosthesis. The hippocampus was chosen because it is thought to be the most ordered and structured part of the brain and is the most studied area. Its function is to encode experiences for storage as long-term memories elsewhere in the brain.

Thomas DeMarse at the University of Florida used a culture of 25,000 neurons taken from a rat's brain to fly a F-22 fighter jet aircraft simulator. After collection, the cortical neurons were cultured in a petri dish and rapidly began to reconnect themselves to form a living neural network. The cells were arranged over a grid of 60 electrodes and used to control the pitch and yaw functions of the simulator. The study's focus was on understanding how the human brain performs and learns computational tasks at a cellular level.

Ethical considerations

There has not been a vigorous debate about the ethical implications of BCIs, even though there are several commercially available systems such as brain pacemakers used to treat neurological conditions, and could theoretically be used to modify other behaviours.

Important topics in the neuroethical debate are: 1) obtaining informed consent from people who have difficulty communicating, 2) risk/benefit analysis 3) shared responsibility of BCI teams (e.g. how to ensure that responsible group decisions can be made), 4) the consequences of BCI technology for the quality of life of patients and their families, 5) side-effects (e.g. neurofeedback of sensorimotor rhythm training is reported to affect sleep quality) 6) personal responsibility and its possible constraints (e.g. who is responsible for erroneous actions with a neuroprosthesis), 7) issues concerning personality and personhood and its possible alteration, 8) therapeutic applications and their possible exceedance, 9) questions of research ethics that arise when progressing from animal experimentation to application in human subjects, 10) mind-reading and privacy, 11) mind-control, 12) selective enhancement and social stratification and 13) communication to the media.

Emory University neuroscience professor Michael Crutcher has expressed concern about BCIs, specifically ear and eye implants: "If only the rich can afford it, it puts everyone else at a disadvantage." Clausen concluded in 2009 that “BCIs pose ethical challenges, but these are conceptually similar to those that bioethicists have addressed for other realms of therapy”. Moreover, he suggests that bioethics is well-prepared to deal with the issues that arise with BCI technologies. Haselager and colleagues pointed out that expectations of BCI efficacy and value play a great role in ethical analysis and the way BCI scientists should approach media. Furthermore, standard protocols can be implemented to ensure ethically sound informed-consent procedures with locked-in patients.

Researchers are well aware that sound ethical guidelines, appropriately moderated enthusiasm in media coverage and education about BCI systems will be of utmost importance for the societal acceptance of this technology. Thus, recently more effort is made inside the BCI community to initiate the development of ethical guidelines for BCI research, development and dissemination.

End-user computing

In computing, **End User Computing** (EUC) refer to systems in which non-programmers can create working applications. EUC is a group of approaches to computing that aim at better integrating end users into the computing environment. These approaches attempt to realize the potential for high-end computing to perform in a trustworthy manner in problem solving of the highest order.

The **EUC Ranges** section describes two types of approaches that are at different ends of a spectrum. A simple example of these two extremes can use the SQL context.

The first approach would have canned queries and reports that for the most part would be invoked with buttons and/or simple commands. In this approach, a computing group would keep these canned routines up to date through the normal development/maintenance methods.
For the second approach, SQL administration would allow for end-user involvement at several levels including administration itself. Users would also define queries though the supporting mechanism may be constrained in order to reduce the likelihood of run-away conditions that would have negative influence on other users. We see this already in some business intelligence methods which build SQL, including new databases, on the fly. Rules might help dampen effects that can occur with the open-ended environment. The process would expect, and accommodate, the possibility of long run times, inconclusive results and such. These types of unknowns are undecidable 'before the fact'; the need to do 'after the fact' evaluation of results is a prime factor of many higher-order computational situations but cannot (will not) be tolerated by an end user in the normal production mode.

Between these two extremes view of **EUC** there are many combinations. Some of the factors contributing to the need for further EUC research are knowledge processing, pervasive computing, issues of ontology, interactive visualization and analysis coupling schemes, and the like.

EUC Ranges

EUC might work by one type of approach that attempts to integrate the human interface ergonomically into a user centered design system throughout its life cycle. In this sense, EUC's goal is to allow unskilled staff to use expensive and highly skilled knowledge in their jobs, by putting the knowledge and expertise into the computer and teaching the end user how to access it. At the same time, this approach is used when highly critical tasks are supported by computational systems (commercial flight, nuclear plant, and the like).

Another approach to EUC allows end users (SMEs, domain experts) to control and even perform software engineering and development. In this case, it can be argued that this type of approach results mainly from deficiencies in computing that could be overcome with better tools and environments. But, high-end roles for the computer in non-trivial domains necessitate (at least, for now) a more full interchange (bandwidth for conversation) that is situational and subject to near exhaustive scrutiny (there are limits influencing how far we can go (bringing up, the necessity for a behavioral framework)). Such cannot be filled by a pre-defined system in today's world. In a sense, the computer needs to have the same credentials as does a cohort (scientific method of peer review) in the discipline. This type of computing falls on the more 'open' side of the fence where scientific knowledge is not wrapped within the cloak of IP.

In the first type of approach of EUC described above, it appears easier to teach factory workers, for example, how to read dials, push buttons, pull levers, and log results than to teach them the manufacturing process and mathematical models. The current computing trend is to simulate a console with similar dials, sliders, levers, and switches, which the end user is taught to use. To further reduce end user training, computer consoles all contain components which are shaped, labeled, coloured, and function similarly. EUC developers assume that once the end user knows what and how a particular lever works, they will quickly identify it when it appears in a new console. This means that once staff learns one console, they will be able to operate all consoles. Admittedly each console will have new components, but training is limited to those, not the whole console. This approach requires more than just Pavlovian responses as the console content will have meaning that is of use and power to the particular computing domain. That is, there may be training that reduces the time between sensor reading and action (such as the situation for a pilot of a commercial plane) however, the meaning behind the reading will include other sensor settings as well as whole context that may be fairly involved.

Computing of this type can be labelled black box where trust will be an essential part, behavioral analysis is the name of the game, and there is a disparate (and very, very wide) gap between the domain and the computer-support ontologies.

In the other type of EUC described above, it has been argued that **a** (teaching programming and computing concepts to a domain expert (say, one of the sciences or engineering disciplines) and letting the expert develop rules (this type of action can be subsumed under the topic of business rules)) is easier than **b** (teaching the intricacies of a complex discipline to a computer worker). **b** is the normal approach of the IT-driven situation. **a** has been the reality since day one of computing in many disciplines. One may further argue that resolving issues of **a** and **b** is not unlike the interplay between distributed and centralized processing (which is an age-old concern in computing). In this sense of EUC, there may be computer scientists supporting decisions about architecture, process, and GUI. However, in many cases, the end user owns the software components. One thrust related to this sense of EUC is a focus on providing better languages to the user. ICAD was an example in the KBE context. Of late, this discipline has moved to a co-joint architecture that features advanced interactive domain visualization coupled with a complicated API accessed via VBA, C++, and the like. This type of co-jointness is an example of a domain tool augmented with non-trivial extensibility.

Trend

The historical view regarding end users is being eroded by the internet and wireless communication, where the traditional end user is able to actively contribute and add value to the computer system.

Another trend is where users specify, and even develop, rules that may be fairly normal relationships (SQL) or be hard-core numerical processes that may require attention being given to serious computational characteristics, such as ill-conditioning, parallelisms. and similar issues of an ongoing nature.

Research

The human interface receives continuous attention as emerging interfaces reveal more possibilities and risks. The quest to both internationalize (i18n) and localize (L10n) software is hampered by computers designed for the English alphabet, but other major languages, such as Chinese, Japanese, and Arabic, have different requirements.

Other studies range from website accessibility to pervasive computing, with the focus ranging from the human to the computer. The issue centres around how much the human can safely and reliably adjust to the computer's I/O devices on the one hand, and how unobtrusively the computer can detect the human's needs on the other.

Furthermore, issues related to computing ontologies (example: the Language/Action perspective has found success in CRM, etc.) continue to be of interest to EUC.

Analysis

The concepts related to the end user cover a wide range, hence **End User Computing** can have a range of forms and values. Most early computer systems were tightly controlled by an IT department; 'users' were just that. However, at any point in the evolution of computer systems through time, there was serious work in several domains that required user development. The dynamics of the power struggles between centralized and decentralized computing have been a fact; this was partially due to the emergence of the mid-sized computers (VAX, etc.). Then, the advent of the personal workstation opened up the door, so to speak, since it allowed a more pervasive type of computation to emerge. The recent advent of 'web' services has extended the issues to a more broad scope.

In the sense of serious domain computing and given the intertwining of computation into all advanced disciplines, any tool (inclusive of any type of capability related to a domain/discipline) that is provided by a computer becomes part of the discipline (methodology, etc.).

As such, the issue arises about how open the tool is to scrutiny. Some disciplines require more understanding of the tool set than do others. That is, tools that are operational in scope require less understanding than those that are ontological. As an example of the latter type of influence on disciplines, consider the impact that the computer has had on the scientific method.

Some of the issues related to End User Computing concern architecture (iconic versus language interface, open versus closed, and others). These continue to be studied. Other issues relate to IP, configuration, maintenance, ... End User Computing allows more user input into system affairs that can range from personalization to full-fledged ownership of the system.

Examples of End User Computing are systems built using the 4GLs, such as MAPPER or SQL, or one of the 5GLs, such as ICAD. ICAD (in the KBE domain) stands as a prime example since it is associated with the pervasive use of Lisp (one of the 3GLs) by Engineers to accomplish remarkable effects through a long economic cycle.

Slogans

- Computing concerns and good End User Computing can be antithetically related.
- Good End User Computing practices might help temper things such as the AI Winter.
- The computational needs to wed with the phenomenal (are 'borgs' inevitable?).
- There is always more than meets the eye (or, GUI, or any interface, is only part of the truth).

Chapter 5

Fitts's Law and GOMS

Fitts's law

Fitts's law (often cited as **Fitts' law**) is a model of human movement in human-computer interaction and ergonomics which predicts that the time required to rapidly move to a target area is a function of the distance to and the size of the target. Fitts's law is used to model the act of *pointing*, either by physically touching an object with a hand or finger, or virtually, by pointing to an object on a computer display using a pointing device. It was proposed by Paul Fitts in 1954.

Model

Fitts's law has been formulated mathematically in several different ways. One common form is the Shannon formulation (proposed by Scott MacKenzie, professor of York University, and named for its resemblance to the Shannon–Hartley theorem) for movement along a single dimension:

$$T = a + b \log_2 \left(1 + \frac{D}{W}\right)$$

where:

- T is the average time taken to complete the movement. (Traditionally, researchers have used the symbol MT for this, to mean *movement time*.)
- a represents the start/stop time of the device (intercept) and b stands for the inherent speed of the device (slope). These constants can be determined experimentally by fitting a straight line to measured data.
- D is the distance from the starting point to the center of the target. (Traditionally, researchers have used the symbol A for this, to mean the *amplitude* of the movement.)
- W is the width of the target measured along the axis of motion. W can also be thought of as the allowed error tolerance in the final position, since the final point of the motion must fall within $\pm W/2$ of the target's center.

From the equation, we see a *speed–accuracy* trade off associated with pointing, whereby targets that are smaller and/or further away require more time to acquire.

Success and implications

Fitts's law is an unusually successful and well-studied model. Experiments that reproduce Fitts's results and/or that demonstrate the applicability of Fitts's law in somewhat different situations are not difficult to perform. The measured data in such experiments often fit a straight line with a correlation coefficient of 0.95 or higher, a sign that the model is very accurate.

Although Fitts only published two articles on his law (Fitts 1954, Fitts and Peterson 1964), there are hundreds of subsequent studies related to it in the human-computer interaction (HCI) literature, and quite possibly thousands of studies published in the larger psychomovement literature. The first HCI application of Fitts's law was by Card, English, and Burr (1978), who used the index of performance (*IP*), defined as $^1/_b$, to compare performance of different input devices, with the mouse coming out on top. This early work, according to Stuart Card's biography, "was a major factor leading to the mouse's commercial introduction by Xerox".

Fitts's law has been shown to apply under a variety of conditions, with many different limbs (hands, feet, head-mounted sights, eye gaze), manipulanda (input devices), physical environments (including underwater), and user populations (young, old, special educational needs, and drugged participants). Note that the constants a, b, *IP* have different values under each of these conditions.

Since the advent of graphical user interfaces, Fitts's law has been applied to tasks where the user must position a mouse cursor over an on-screen target, such as a button or other widget. Fitts's law models both point-and-click and drag-and-drop actions. Dragging has a lower *IP* associated with it, because the increased muscle tension makes pointing more difficult. Fitts's law has also been shown to model target-directed hand and head motions in a virtual environment.

In its original and strictest form:

- It applies only to movement in a single dimension and not to movement in two dimensions (though it is successfully extended to two dimensions in the Accot-Zhai steering law);
- It describes simple motor response of, say, the human hand, failing to account for software acceleration usually implemented for a mouse cursor;
- It describes untrained movements, not movements that are executed after months or years of practice (though some argue that Fitts's law models behaviour that is so low level that extensive training doesn't make much difference).

If, as generally claimed, the law does hold true for pointing with the mouse, some consequences for user-interface design include:

- Buttons and other GUI controls should be a reasonable size; it is relatively difficult to click on small ones.
- Edges and corners of the computer display (e.g., the location of the Start button in Microsoft Windows and the menus and Dock of Mac OS X) are particularly easy to acquire because the pointer remains at the screen edge regardless of how much further the mouse is moved, thus can be considered as having infinite width.
- Similarly, top-of-screen menus (e.g., Mac OS) are easier to acquire than top-of-window menus (e.g., Windows OS).
- Pop-up menus can usually be opened faster than pull-down menus, since the user avoids travel: the pop-up appears at the current cursor position.
- Pie menu items typically are selected faster and have a lower error rate than linear menu items, for two reasons: because pie menu items are all the same, small distance from the centre of the menu; and because their wedge-shaped target areas (which usually extend to the edge of the screen) are very large.

Fitts's law remains one of the few hard, reliable human-computer interaction predictive models, joined more recently by the Accot-Zhai steering law, which is derived from Fitts's law.

Mathematical details

The logarithm in Fitts's law is called the index of difficulty *ID* for the target, and has units of bits. We can rewrite the law as

$$T = a + bID,$$

where

$$ID = \log_2 \left(\frac{D}{W} + 1 \right).$$

Thus, the units for *b* are time/bit; e.g., ms/bit. The constant *a* can be thought of as incorporating reaction time and/or the time required to click a button.

The values for *a* and *b* change as the conditions under which pointing is done are changed. For example, a mouse and stylus may both be used for pointing, but have different constants *a* and *b* associated with them.

An index of performance *IP* (also called throughput *TP*), in bits/time, can be defined to characterize how quickly pointing can be done, independent of the particular targets involved. There are two conventions for defining *IP*: one is $IP = 1/b$ (which has the disadvantage of ignoring the effect of *a*), the other is $IP = ID_{\text{average}}/MT_{\text{average}}$ (which has the disadvantage of depending on an arbitrarily chosen "average" *ID*). Whatever definition is used, measuring the *IP* of different input devices allows the devices to be compared with respect to their pointing capability.

Slightly different from the Shannon formulation is the original formulation by Fitts:

$$ID = \log_2 \left(\frac{2D}{W} \right).$$

The factor of 2 here is not particularly important; this form of the *ID* can be rewritten with the factor of 2 absorbed as changes in the constants *a*, *b*. The "+1" in the Shannon form, however, does make it different from Fitts's original form, especially for low values of the ratio *D*/*W*. The Shannon form has the advantage that the *ID* is always non-negative, and has been shown to better fit measured data.

Derivation

Fitts's law can be derived from various models of motion. A very simple model, involving discrete, deterministic responses, is considered here. Although this model is overly simple, it provides some intuition for Fitts's law.

Assume that the user moves toward the target in a sequence of submovements. Each submovement requires a constant time *t* to execute, and moves a constant fraction 1-*r* of the remaining distance to the centre of the target, where $0 < r < 1$. Thus, if the user is initially at a distance *D* from the target, the remaining distance after the first submovement is *rD*, and the remaining distance after the *n*th submovement is $r^n D$. (In other words, the distance left to the target's centre is a function that decays exponentially over time.) Let *N* be the (possibly fractional) number of submovements required to fall within the target. Then,

$$r^N D = \frac{W}{2}.$$

Solving for *N*:

$$\begin{aligned} N &= \log_r \frac{W}{2D} \\ &= \frac{1}{\log_2 r} \log_2 \frac{W}{2D} \quad (\text{since } \log_x y = (\log_z y)/(\log_z x)) \\ &= \frac{1}{\log_2 1/r} \log_2 \frac{2D}{W} \quad (\text{since } \log_x y = -\log_x (1/y)). \end{aligned}$$

The time required for all submovements is:

$$T = Nt = \frac{t}{\log_2 1/r} \log_2 \frac{2D}{W}$$
$$= \frac{t}{\log_2 1/r} + \frac{t}{\log_2 1/r} \log_2 \frac{D}{W}.$$

By defining appropriate constants *a* and *b*, this can be rewritten as

$$T = a + b \log_2 \frac{D}{W}.$$

The above derivation is similar to one given in Card, Moran, and Newell (1983).

GOMS

GOMS (**Goals, Operators, Methods, and Selection rules**) is a kind of specialized human information processor model for human computer interaction observation. Developed in 1983 by Stuart Card, Thomas P. Moran and Allen Newell, it was explained in their book *The Psychology of Human Computer Interaction*. Following these initial steps, additional models for analysis evolved and are heavily used in the engineering-oriented usability community.

Overview

GOMS reduces a user's interaction with a computer to its elementary actions (these actions can be physical, cognitive or perceptual). Using these elementary actions as a framework an interface can be studied. There are several different GOMS variations which allow for different aspects of an interface to be accurately studied and predicted.

For all of the variants, the definitions of the major concepts are the same. **Goals** are what the user intends to accomplish. **Operators** are actions that are performed to get to the goal. **Methods** are sequences of operators that accomplish a goal. There can be more than one method available to accomplish a single goal, if this is the case then **selection rules** are used to describe when a user would select a certain method over the others. Selection rules are often ignored in typical GOMS analyses. There is some flexibility for the designers/analysts definition of all of these entities. For instance, one person's operator may be another’s goal. The level of granularity is adjusted to capture what the particular evaluator is examining!

Advantages of GOMS Overall

The GOMS method is not necessarily the most accurate of human-computer interface interaction measurement methods, but it certainly has its advantages. A GOMS estimate of a particular interaction can be calculated with little effort, at little cost, and in a short amount of time if the average Methods-Time Measurement data for each specific task has been previously measured experimentally to a high degree of accuracy. With a careful investigation into all of the detailed steps necessary for a user to successfully interact with an interface, the time measurement of how long it will take a user to interact with that interface is a simple calculation. Summing the times necessary to complete the detailed steps provides an estimate for how long it will take a user to successfully complete the desired task.

Weaknesses of GOMS Overall

All of the GOMS techniques provide valuable information, but they all also have certain drawbacks. None of the techniques address user unpredictability - such as user behaviour being affected by fatigue, social surroundings, or organizational factors. The techniques are very explicit about basic movement operations, but are generally less rigid with basic cognitive actions. It is a fact that slips cannot be prevented, but none of the GOMS models allow for any type of error. Further, all of the techniques work under the assumption that a user will know what to do at any given point - so they apply only to expert users, not novices.

Functionality of the system is not considered, only the usability. If functionality were considered, the evaluation could make recommendations as to which functions should be performed by the system (i.e. mouse snap). User personalities, habits or physical restrictions (for example disabilities) are not accounted for in any of the GOMS models. All users are assumed to be exactly the same. Recently some extensions of GOMS were developed, that allow to formulate GOMS models describing the interaction behavior of disabled users .

Except for KLM (Keystroke Level Modeling), the evaluators are required to have a fairly deep understanding of the theoretical foundations of GOMS, CCT (Cognitive Complexity Theory), or MHP (Model Human Processor). This limits the effective use of GOMS to large entities with the financial power to hire a dedicated human computer interaction (HCI) specialist or contract with a consultant with such expertise.

Variations

The plain, or "vanilla flavored" GOMS first introduced by Card, Moran and Newell is now referred to as CMN-GOMS. Keystroke Level Modeling (KLM) is the next GOMS technique and was also introduced by Card, Moran and Newell in their 1983 book. This technique makes several simplifying assumptions that make it really just a restricted version of GOMS. The third major variant on the GOMS technique is the 'Natural GOMS Language' or NGOMSL. This technique gives a very strict, but natural, language for building GOMS models. The final variation of GOMS is CPM-GOMS. This technique is based on the Model Human Processor. The main advantage of CPM-GOMS is that it allows for the modeling of parallel information processing by the user, however it is also the most difficult GOMS technique to implement.

Summary of CMN-GOMS Application

The CMN-GOMS method assumes that information is comprehended by a user in the following manner:

- Eyes/ears perceive information
- Information enters perceptual processor
- Information enters the visual/auditory image store
- Information is stored in the working memory and long term memory
- Information is analyzed in the cognitive processor and a desired reaction (motor function) is chosen
- Desired motor function is activated in the motor processor
- Desired motor function is applied by user's body

All measurements are provided in the following form: middleman[fastman, slowman]. The "middleman" term is the most typical time it would take to complete the action, or the time that is most representative of the average user (not the mean, average, or median, but the mode: the time that is most often measured). The fastman is a "best case" scenario. It is the reasonably best possible statistic. Note that, despite the name, it is not necessarily always the fastest time. It is instead the time that is expected to be the best a user could possibly do. The slowman time is, contrarily, a "worst case scenario."

In CMN-GOMS, the following Methods-Time Measurement data should be used:

- Eye fixation = 230[70, 700] milliseconds
- Eye movement = 30 milliseconds
- Perceptual Processor = 100[50, 200] milliseconds
- Cognitive Processor = 70[25, 170] milliseconds
- Motor Processor = 70[30, 100] milliseconds

Also important in CMN-GOMS is the time it takes to apply the motor function once it is processed. For this, a user can apply Fitt's Law.

Summary of KLM Application

The Keystroke Level Model is a less accurate, but faster application than CMN-GOMS. It is especially useful when determining time it takes to type a phrase, correct a realized error, or select something with a mouse. It uses the following average times as measured by Card, Moran and Newell:

- Press a key or button

    - Best typist = .08 seconds
    - Good typist = .12 seconds
    - Average skilled typist = .20 seconds
    - Average non-secretary = .28 seconds

    - Typing random letters = .50 seconds
    - Typing complex codes = .75 seconds
    - Worst typist = 1.2 seconds

- Point with a mouse (excluding click) = 1.1 seconds
- Move hands to keyboard from mouse (or vice-versa) = .4 seconds
- Mentally prepare = 1.35 seconds

Typing a word, assuming a subject's hands are already on the keyboard, would therefore be calculated by multiplying the number of letters in the word by the value given above to "press a key or button." Note that categorizing the subject into an accurate typing skill level impacts the estimated measurement greatly.

Importance of Assumptions in GOMS Analysis

Accurate assumptions are vital in GOMS analysis. Before applying the average times for detailed functions, it is very important that an experimenter make sure he or she has accounted for as many variables as possible by using assumptions. Experimenters should design their GOMS analysis for the users who will most likely be using the system which is being analyzed. Consider, for example, an experimenter wishes to determine how long it will take an F22 Raptor pilot to interact with an interface he or she has used for years. It can probably be assumed that the pilot has outstanding vision and is in good physical health. In addition, it can be assumed that the pilot can interact with the interface quickly because of the vast hours of simulation and previous use he or she has endured. All things considered, it is fair to use fastman times in this situation. Contrarily, consider an 80-year-old woman with no flight experience attempting to interact with the same F22 Raptor interface. It is fair to say that the two people would have much different skill sets and those skill sets should be accounted for subjectively.

Accounting for Errors

The only way to account for errors in GOMS analysis is to predict where the errors are most likely to occur and measure the time it would take to correct the predicted errors. For example, assume an experimenter thought that in typing the word "the" it was likely that a subject would instead incorrectly type "hte." The experimenter would calculate the time it takes to type the incorrect word, the time it takes to recognize that a mistake has been made, and the time it takes to correct the recognized error.

An experimenter should not, however, assume that an error will occur every time a subject does an action. James Reason calculated probabilities that an error will occur. According to Reason, a skill error is defined as an unconscious, automatic action resulting in an error (for example a mistyped key, a key hit the wrong number of times, a skipped key, etc.). A skill error will occur with a probability of .006 for young users and .011 for old users. A rule error, contrarily, is defined as following a series of steps and either making a mistake applying good rules incorrectly or applying bad rules at wrong times. Simple rule errors occur with a probability of .036 for young users and .024 for old users. Complex rule errors occur with a probability of .156 for young users and .324 for old users.

A concrete application of this idea is a GOMS model for keyboard navigation in Web pages. This model contains a probability for a focus loss during navigation inside the page using the TAB key.

Successful Applications of GOMS

A successful implementation of CPM-GOMS was in *Project Ernestine* held by New England Telephone. New ergonomically designed workstations were compared to old workstations in terms of improvement in telephone operators' performance. CPM-GOMS analysis estimated a 3% decrease in productivity. Over the four month trial 78,240 calls were analysed and it was concluded that the new workstations produced an actual 4% decrease in productivity. As the proposed workstation required less keystrokes than the original it was not clear from the time trials why the decrease occurred. However CPM-GOMS analysis made it apparent that the problem was that the new workstations did not utilize the workers' slack time. Not only did CPM-GOMS give a close estimate, but it provided more information of the situation.

Chapter 6

Human Information Processor Model and Human–Computer Information Retrieval

Human information processor model

Human processor model is a cognitive modeling method used to calculate how long it takes to perform a certain task. Other cognitive modeling methods include parallel design, GOMS, and KLM (human-computer interaction). Cognitive modeling methods are one way to evaluate the usability of a product. This method uses experimental times to calculate cognitive and motor processing time. The value of the human processor model is that it allows a system designer to predict the performance with respect to time it takes a person to complete a task without performing experiments. Other modeling methods include inspection methods, inquiry methods, prototyping methods, and testing methods.

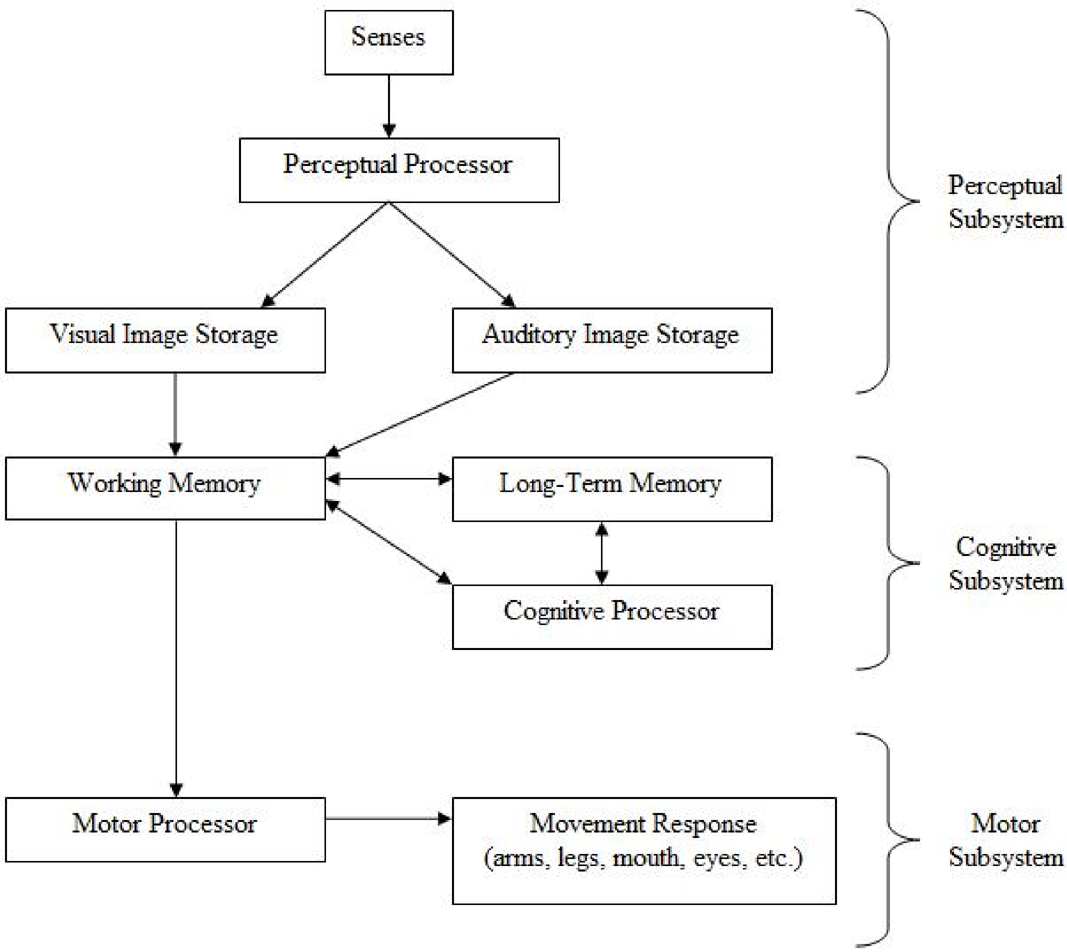
The human processor model uses the cognitive, perceptual, and motor processors along with the visual image, working memory, and long term memory storages. A diagram is shown below. Each processor has a cycle time and each memory has a decay time. These values are also included below. By following the connections diagrammed below, along with the associated cycle or decay times, the time it takes a user to perform a certain task can be calculated. Studies into this field were initially done by Card, S.K., Moran T.P., & Newell, A. (1983). Current studies in the field include work to distinguish process times in older adults by Tiffany Jastrembski and Neil Charness (2007).

How To Calculate

The calculations depend on the ability to break down every step of a task into the basic process level. The more detailed the analysis the more accurate the model will be to predict human performance. The method for determining processes can be broken down into the following steps.

- Write out main steps based on: a working prototype, simulation, step by step walk-through of all steps
- Clearly identify the specific task and method to accomplish that task

- For each final step identify sub-levels down to a basic process (in the diagram or chart below)
- Convert into pseudo code (writing out methods for each step)
- List all assumptions (will be helpful as multiple iterations are completed)
- Determine time of each operation (based on the table below)
- Determine if operation times should be adjusted (slower for elderly, disability, unfamiliarity, etc.)
- Sum up execution times
- Iterate as needed and check with prototyping if possible



Parameter	Mean	Range
Eye movement time	230 ms	70-700 ms
Decay half-life of visual image storage	200 ms	90-1000 ms
Visual Capacity	17 letters	7-17 letters
Decay half-life of auditory storage	1500 ms	90-3500 ms
Auditory Capacity	5 letters	4.4-6.2 letters

Perceptual processor cycle time	100 ms	50-200 ms
Cognitive processor cycle time	70 ms	25-170 ms
Motor processor cycle time	70 ms	30-100 ms
Effective working memory capacity	7 chunks	5-9 chunks
Pure working memory capacity	3 chunks	2.5-4.2 chunks
Decay half-life of working memory	7 sec	5-226 sec
Decay half-life of 1 chunk working memory	73 sec	73-226 sec
Decay half-life of 3 chunks working memory	7 sec	5-34 sec

Potential Uses

Once complete, the calculations can then be used to determine the probability of a user remembering an item that may have been encountered in the process. The following formula can be used to find the probability: $P = e^{-K*t}$ where K is the decay constant for the respective memory in question (working or long term) and t is the amount of time elapsed (with units corresponding to that of K). The probability could then be used to determine whether or not a user would be likely to recall an important piece of information they were presented with while doing an activity.

It is important to deduce beforehand whether the user would be able to repeat the vital information throughout time t, as this has a negative impact on the working memory if they cannot. For example, if a user is reading lines of text and is presented with an important phone number in that text, they may not be able to repeat the number if they have to continue to read. This would cause the user's working memory's decay time to be smaller, thus reducing their probability of recall.

Human–computer information retrieval

Human–computer information retrieval (**HCIR**) is the study of information retrieval techniques that bring human intelligence into the search process. The fields of human–computer interaction (HCI) and information retrieval (IR) have both developed innovative techniques to address the challenge of navigating complex information spaces, but their insights have often failed to cross disciplinary borders. Human–computer information retrieval has emerged in academic research and industry practice to bring together research in the fields of IR and HCI, in order to create new kinds of search systems that depend on continuous human control of the search process.

History

This term *human–computer information retrieval* was coined by Gary Marchionini in a series of lectures delivered between 2004 and 2006. Marchionini's main thesis is that

"HCIR aims to empower people to explore large-scale information bases but demands that people also take responsibility for this control by expending cognitive and physical energy."

In 1996 and 1998, a pair of workshops at the University of Glasgow on information retrieval and human–computer interaction sought to address the overlap between these two fields. Marchionini notes the impact of the World Wide Web and the sudden increase in information literacy – changes that were only embryonic in the late 1990s.

A few workshops have focused on the intersection of IR and HCI. The Workshop on Exploratory Search, initiated by the University of Maryland Human-Computer Interaction Lab in 2005, alternates between the Association for Computing Machinery Special Interest Group on Information Retrieval (SIGIR) and Special Interest Group on Computer-Human Interaction (CHI) conferences. Also in 2005, the European Science Foundation held an Exploratory Workshop on Information Retrieval in Context. Then, the first Workshop on Human Computer Information Retrieval was held in 2007 at the Massachusetts Institute of Technology.

What is HCIR?

HCIR includes various aspects of IR and HCI. These include exploratory search, in which users generally combine querying and browsing strategies to foster learning and investigation; information retrieval in context (i.e., taking into account aspects of the user or environment that are typically not reflected in a query); and interactive information retrieval, which Peter Ingwersen defines as "the interactive communication processes that occur during the retrieval of information by involving all the major participants in information retrieval (IR), i.e. the user, the intermediary, and the IR system."

A key concern of HCIR is that IR systems intended for human users be implemented and evaluated in a way that reflects the needs of those users.

Most modern IR systems employ a ranked retrieval model, in which the documents are scored based on the probability of the document's relevance to the query. In this model, the system only presents the top-ranked documents to the user. This systems are typically evaluated based on their mean average precision over a set of benchmark queries from organizations like the Text Retrieval Conference (TREC).

Because of its emphasis in using human intelligence in the information retrieval process, HCIR requires different evaluation models – one that combines evaluation of the IR and HCI components of the system. A key area of research in HCIR involves evaluation of these systems. Early work on interactive information retrieval, such as Juergen Koenemann and Nicholas J. Belkin's 1996 study of different levels of interaction for automatic query reformulation, leverage the standard IR measures of precision and recall but apply them to the results of multiple iterations of user interaction, rather than to a single query response. Other HCIR research, such as Pia Borlund's IIR evaluation model, applies a methodology more reminiscent of HCI, focusing on the characteristics of users, the details of experimental design, etc.

Goals

Marchionini put forth the following goals towards a system where the user has more control in determining relevant results:

- Systems should aim to get people closer to the information they need, especially to the meaning; that is, systems can no longer only deliver the relevant documents, but must also provide facilities for making meaning with those documents.

- Systems should increase user responsibility as well as control; that is, information systems require human intellectual effort, and good effort is rewarded.

- Systems should have flexible architectures so they may evolve and adapt to increasingly more demanding and knowledgeable installed bases of users over time.

- Systems should aim to be part of information ecology of personal and shared memories and tools rather than discrete standalone services.

- Systems should support the entire information life cycle (from creation to preservation) rather than only the dissemination or use phase.

- Systems should support tuning by end users and especially by information professionals who add value to information resources.

- Systems should be engaging and fun to use.

Techniques

The techniques associated with HCIR emphasize representations of information that use human intelligence to lead the user to relevant results. These techniques also strive to allow users to explore and digest the dataset without penalty, i.e., without expending unnecessary costs of time, mouse clicks, or context shift.

Many search engines have features that incorporate HCIR techniques. Spelling suggestions and automatic query reformulation provide mechanisms for suggesting potential search paths that can lead the user to relevant results. These suggestions are presented to the user, putting control of selection and interpretation in the user’s hands.

Faceted search enables users to navigate information hierarchically, going from a category to its sub-categories, but choosing the order in which the categories are presented. This contrasts with traditional taxonomies in which the hierarchy of categories is fixed and unchanging. Faceted navigation, like taxonomic navigation, guides users by showing them available categories (or facets), but does not require them to browse through a hierarchy that may not precisely suit their needs or way of thinking.

Lookahead provides a general approach to penalty-free exploration. For example, various web applications employ AJAX to automatically complete query terms and suggest popular searches. Another common example of lookahead is the way in which search engines annotate results with summary information about those results, including both static information (e.g., metadata about the objects) and "snippets" of document text that are most pertinent to the words in the search query.

Relevance feedback allows users to guide an IR system by indicating whether particular results are more or less relevant.

Summarization and analytics help users digest the results that come back from the query. Summarization here is intended to encompass any means of aggregating or compressing the query results into a more human-consumable form. Faceted search, described above, is one such form of summarization. Another is clustering, which analyzes a set of documents by grouping similar or co-occurring documents or terms. Clustering allows the results to be partitioned into groups of related documents. For example, a search for "java" might return clusters for Java (programming language), Java (island), or Java (coffee).

Visual representation of data is also considered a key aspect of HCIR. The representation of summarization or analytics may be displayed as tables, charts, or summaries of aggregated data. Other kinds of information visualization that allow users access to summary views of search results include tag clouds and treemapping.

Chapter 7

IDrive and Keystroke-Level Model

iDrive

iDrive is a computer system which is used to control most secondary vehicle systems in many current BMW cars. iDrive's user interface consists of a LCD panel mounted in the dashboard and a controller knob mounted on the center console.

iDrive allows the driver and front-seat passenger (in recent cars it is available to back-seat passengers as well) to control such amenities as the climate (air conditioner and heater), the audio system (radio and CD player), the navigation system and communication system.

iDrive Generations

iDrive (1st Gen)

Debuted in September 2001 with the 7-Series and was based on Microsoft Windows CE for Automotive.
This can easily be seen when the system reboots or restarts after a software crash displaying a "Windows CE" logo.

iDrive Business (M-ASK)

M-ASK stands for MMI Audio System Kontroller and is manufactured by Becker. This is a limited version of the iDrive computer with a small 6.6" display and is only found on 5, 6 and 7-series, and the X5 or X6, without the navigation option.

In addition it can optionally be ordered in Europe on the 1-series and 3-series as "Business navigation", which has basic navigation abilities. Early versions of the Business navigation could only display directional arrows, but the latest version can also display 2D maps. iDrive Business Navigation uses a different map DVD then iDrive Professional Navigation. In addition because only one optical drive is available you cannot use both navigation and listen to a CD simultaneously.

When iDrive Professional is ordered the M-ASK system is replaced by either the CCC or CIC in addition to a larger display.

iDrive Business is available on the following cars;

- iDrive Business Navigation (optional)
  - 1-Series E81/E82/E87/E88
  - 3-Series E90/E91/E92/E93
- iDrive Business (default when navigation is not ordered)
  - 5-Series E60/E61
  - 6-Series E63/E64
  - 7-Series E65/E66/E67/E68
  - 7-Series F01/F02
  - X5 E70
  - X6 E71

Note that the above list can vary depending on the region. For instance North American 7-series come by default with CCC or CIC.

iDrive Professional Navigation (CCC)

iDrive Controller on BMW X5

iDrive controller on BMW 530d

Debuted in 2003 with the E60/E61 5-Series and is based on Wind River VxWorks, a real-time operating system. CCC stands for Car Communication Computer and uses a larger 8.8" wide-screen display. It was available on the following cars as an option;

- 1-Series E81/E82/E87/E88 manufactured until March 2007
- 3-Series E90/E91/E92/E93 manufactured until March 2007
- 5-Series E60/E61 manufactured until March 2007
- 6-Series E63/E64 manufactured until March 2007
- 7-Series E65/E66/E67/E68 from September 2004

CCC based systems use a map DVD from Navteq in a dedicated DVD drive.

CCC - update 1

This is a minor update to iDrive Professional debuted in March 2007. It adds additional programmable buttons in the dashboard to directly access frequent functions and it removes the haptic feedback from the iDrive controller. It is available on the following cars as an option;

- 1-Series E81/E82/E87/E88 manufactured between March 2007 and September 2008
- 3-Series E90/E91/E92/E93 manufactured between March 2007 and September 2008
- 5-Series E60/E61 manufactured between March 2007 and March 2010
- 6-Series E63/E64 manufactured between March 2007 and September 2009
- X5 E70
- X6 E71

CCC - update 2

This is a minor update debuted in September 2008 to iDrive Professional equipped cars that did not get the new CIC based system. These cars get the new iDrive controller that is also used on cars with CIC. The actual iDrive computer (CCC) remains the same. This update is available on the following cars;

- 5-Series E60/E61 manufactured from September 2008
- 6-Series E63/E64 manufactured from September 2008

iDrive Professional Navigation (CIC)

Debuted in September 2008 with F01/F02 7-Series. CIC stands for Car Infotainment Computer and is manufactured by Becker, utilizing the QNX operating system. It is available on the following cars as an option;

- 1-Series E81/E82/E87/E88 manufactured from September 2008
- 3-Series E90/E91/E92/E93 manufactured from September 2008
- 5-Series F07/F10/F11 manufactured from September 2009
- 6-Series E63/E64 manufactured from September 2009
- 7-Series F01/F02 manufactured from September 2008

The CIC system is a major update to iDrive, replacing the display, computer and the controller. The display is of a higher resolution, and is generally more responsive than CCC, to address one of the common complaints of iDrive. Internet access is also supported.

CIC-based systems use maps from TeleAtlas which are installed on an internal 2.5" 40GB Hard Disk Drive (HDD). This HDD can also store up to 8GB of music files for playback. For facilitating the uploading of music files to the HDD, a USB port is provided in the glove box.

Following 2009 LCI production, all CIC-based iDrive systems will support DVD video. This, however, is only operational when the vehicle is in the "Park" position.

Rationale

The design rationale of iDrive is to replace a confusing array of controls for the above systems with an all-in-one unit. The controls necessary for vehicle control and safety, such as the headlights and turn signals, are still located in the immediate vicinity of the steering column. iDrive also allows the On-Board Diagnostics computer to provide detailed information to the driver and service technicians in plain-text, rather than limited and confusing symbology, such as a "Check Engine" light.

Since the climate, audio, navigation and communication systems are adjusted only occasionally, they were moved into an easy-to-use, central location. However, iDrive was filled with controversy and many disapproved of it, as it was considered to be difficult to use .

Controversy and Critics

iDrive has caused significant controversy among users, the automotive media, and critics. Many reviewers of BMWs in (automobile) magazines disapprove of the system. Criticisms of iDrive include its steep learning curve and its tendency to cause the driver to look away from the road too much . Most users report that they adapt to the system after about one hour of practice, and the advent of voice controls has reduced the learning curve greatly.

A new iDrive system (CIC) was introduced in September 2008 to address most of the complaints.

Influence

Despite the criticisms of iDrive, the concept of a computer-oriented interface has grown in the luxury segment of the industry:

- MMI, Audi
- Comand APS, Mercedes-Benz
- Remote Touch, Lexus

As such, the significance of iDrive in auto history is predicated more on its pioneering value as a first mover into a new feature space than on the success of its application or particular interpretation of how to execute the idea in design. Generally speaking, the single touch-point for an increasingly broad array of controls is becoming a standard functionality and, like other features that were once only found in luxury vehicles (power windows, air conditioning, air bags) this single touch-point, if it remains a strong selling feature, will likely trickle down to higher-production, more affordable model lines.

How It Works

The *iDrive* M-ASK and CCC systems are based around the points of a compass (north, south, east, west) with each direction corresponding with a specific area. These areas are also colour coded providing identification as to which part of the system is currently being viewed.

- North (blue) for communication
- East (green) for navigation (In some models without navigation, this option is replaced by the On Board Computer)
- South (yellow) for entertainment
- West (red) for climate control

Also, at the iDrive home screen, the control knob can be pushed downwards to provide access to the car's control panel (iMenu), which gives access to a variety of operational parameters:

- date/time settings
- System language localization (generally, English (US/UK) and native language of the vehicle's location is provided.)
- Bluetooth settings
- Flat tire sensor settings,
- lighting behavior, etc.
- custom steering wheel buttons
- Power Output (BMW M series, for example, 400 hp, 500 hp).

Starting in 2007, perhaps in response to critics, iDrive added programmable buttons (6 USA/Japan, 8 in Europe) to the dashboard, breaking tradition of having the entire system operated via the control knob. Each button can be programmed to instantly access any feature within iDrive (such as a particular navigation route, or your favorite radio station). In addition, a dedicated AM/FM button, and a Mode button (to switch between entertainment sources) were added (only USA)

iDrive is controlled by manipulating the control knob. This can be pushed north, south, east, west; pushed inwards (down), nudged left and right, or rotated left and right. The control knob contains an element of 'force feedback' so the knob can not be rotated further than there are options on the screen. It will also kick back slightly as it scrolls through options. This makes it easier to use while driving and keeping your eyes on the road.

From the home screen, each option can be reached by pushing the control knob in the desired direction. From any other screen, pushing the knob in the desired direction and holding in place for a few seconds will jump to that new area. (For example, in the *Entertainment* screen, pushing the control knob to the right for a few seconds before letting it go will jump straight to the *Navigation* area). The iDrive system includes a button marked *Menu* behind the control knob which provides quick access to the home screen.

iDrive is unique among its competitors by using a widescreen display that is split into a 2/3 main window, and 1/3 "Assistance Window". This allows the driver to use any function or menu, while simultaneously maintaining secondary information. For example, if the driver is not in the Navigation menu, he can still see a map on the assistance window. Other information that can be displayed includes navigation route directions and a trip computer.

The latest generation of the BMW iDrive in the 2009 BMW 7-Series features a large 10.2 inch high resolution LCD panel with a 1280x480 resolution. It has a full featured web browser that connects to the internet via BMW ConnectedDrive using EDGE.

Keystroke-Level Model

Keystroke-Level Model, sometimes referred to as KLM or KLM-GOMS, is a hard science approach to human–computer interaction (HCI), based on CMN-GOMS, developed by Card, Moran, and Newell, and spelled out in their book *The Psychology of Human-Computer Interaction*, 1983. The Keystroke-Level Model is an 11-step method that can be used by individuals or companies seeking ways to estimate the time it takes to complete simple data input tasks using a computer and mouse. By using KLM-GOMS, individuals often find more efficient or better ways to complete a task simply by analyzing the steps required in the process and rearranging or eliminating unneeded steps.

It is designed to be easier to use than other GOMS methods, such that companies who cannot afford human–computer interaction specialists can use it. KLM-GOMS is usually applied in situations that require minimal amounts of work and interaction with a computer interface or software design. The calculations and the number of steps required to accurately compute the overall task time increase quickly as the number of tasks involved increases. Thus, KLM-GOMS is best suited to evaluate and time specific tasks that require, on average, less than 5 minutes to complete.

How does it work

The KLM-GOMS model is designed to be as straightforward as possible. The sequence of operations is modeled as a sequence of a small number of operations. Each operation is assigned a duration, which is intended to model the average amount of time an experienced user would take to perform it.

Step Terminology

Kieras (1993, 2001) defines the following operations:

- **K**, key press and release (keyboard)

```
          Best Typist (135 wpm)                        ---          0.08
seconds
          Good Typist (90 wpm)                         ---          0.12
seconds
          Poor Typist (40 wpm)                         ---          0.28
seconds
          Average Skilled Typist (55 wpm)              ---          0.20
seconds
          Average Non-secretary Typist (40 wpm)        ---          0.28
seconds
          Typing Random Letters                        ---          0.50
seconds
          Typing Complex Codes                         ---          0.75
seconds
          Worst Typist (unfamiliar with keyboard)      ---          1.20
seconds
```

- **P**, Point the mouse to an object on screen --- 1.10 seconds
- **B**, Button press or release (mouse) --- 0.10 seconds
- **H**, Hand from keyboard to mouse or vice versa --- 0.40 seconds
- **M**, Mental Preparation --- 1.20 seconds
- **T(n)**, Type string of characters (n * K sec.)
- **W(t)**, User waiting for the system to respond
- **D(n0,l0)**, Draw n0 straight line segment with mouse of total length l0 centimeters

A mouse click would be written "BB" (button press, button release) while a sequence of 3 keyboard key presses is "AAA" or "T(n)" ("T" stands for "type").

A common operation that involves pointing and clicking something on the screen would be written "MPBB" (think, point, press, release).

KLM is not the only technique for evaluating interfaces, but it can be used to compare the speed of two different interfaces designed to accomplish the same task.

Testing Steps

The following 11 steps were developed based upon the methodologies and ideas of Kieras and Olson-Olson and describe the proper approach to calculate the time it takes to complete a task by using a computer interface and hardware.

Step 1--Obtain a working prototype of computer interface or a step by step operational description of a task.

Step 2--Identify the goals or the desired outcome of work .

Step 3--For each of these goals, find subgoals or tasks that achieve the main goals.

Step 4--Identify methods to main goals and all subgoals.

Step 5--Convert description of methods to pseudo-code (the terminology that is described above).

Step 6--State any and all assumptions used in the making of pseudo-code and goals.

Step 7--Determine appropriate mental or keystroke operators for each step.

Step 8--Assign time values to mental or keystroke operators.

Step 9--Add up execution times for operators.

Step 10-Adjust total time of task to be sensitive by age of expected.

Age	Time Adjustment Multiplier (Multiply task time by this value)
Use as is	
30-40	???
40-55	By 1.4
55-60	By 1.7
> 65	By 2.2

Step 11-Verify validity of results

Example Calculation

Change the code in Line 512 from "hte" to "the" ('the'the' in computer language) and then press Enter Key

or

" Change 512 'hte'the' "

Steps

- Mentally prepare
- 6 average speed keystrokes (for word "change")
- 1 average speed keystrokes for space bar
- 3 keystrokes for pressing numbers "512"
- 1 average speed keystrokes for space bar
- Mentally Prepare by reading what needs to be changed
- 6 average keystrokes for typing hte and the
- 3 keystrokes to press the 3 " ' "
- 1 keystroke to press Return Key

Values

K1 = average speed keystrokes (0.20 seconds)

K2 = typing random letters (0.50 seconds)

M = Mentally Prepare (1.35 seconds)

Put above steps into formula: M + 6K1 + K1 + 3K2 + K2 + M + 3K1 + K2 + 3K1 + K2 = 2M + 14K1 + 6K2 = **9.82 seconds**

This method assumes that operator times are invariant and do not depend on the previous sequence of events. New physical operators can be added if their timing can be represented as a simple context-free function. KLM-GOMS does not account for either slips or mistakes automatically—the analyst must create separate models of error sequences and perform their own sensitivity analysis.

The placement of the keystrokes and pointer operations are straightforward, but the placement of the mental operations is not. Mental operations are placed by a set of rules that require some interpretation, such as determining a conceptual "cognitive unit" or grouping of actions. For instance, pressing ctrl and c simultaneously to perform a "copy" would be considered a single cognitive unit. Mental operations are inserted before each cognitive unit to account for cognitive preparation and decision-making.

The main reason a designer or analysts would use this technique is that it is a very fast. Different designs or systems can be compared against one another quickly. It does not require that the evaluator be an expert in GOMS because the procedure is an explicitly laid out recipe. A major caution is that the algorithm is designed to estimate the execution time for an expert user, which is typically faster than the time for a new user or an unfamiliar task.

Chapter 8

Usability Engineering

Usability engineering is a field that is concerned generally with human-computer interaction and specifically with making human-computer interfaces that have high usability or user friendliness. In effect, a user-friendly interface is one that allows users to effectively and efficiently accomplish the tasks for which it was designed and one that users rate positively on opinion or emotional scales. Assessing the usability of an interface and recommending ways to improve it is the purview of the Usability Engineer. The largest subsets of Usability Engineers work to improve usability of software graphical user interfaces (GUIs), web-based user interfaces, and voice user interfaces (VUIs).

Several broad disciplines including Psychology, Human Factors and Cognitive Science subsume usability engineering, but the theoretical foundations of the field come from more specific domains: human perception and action; human cognition; behavioral research methodologies; and, to a lesser extent, quantitative and statistical analysis techniques.

When usability engineering began to emerge as a distinct area of professional practice in the mid- to late 1980s, many usability engineers had a background in Computer Science or in a sub-field of Psychology such as Perception, Cognition or Human Factors. Today, these academic areas still serve as springboards for the professional practitioner of usability engineering, but Cognitive Science departments and academic programs in Human-Computer Interaction now also produce their share of practitioners in the field.

The term **usability engineering** (in contrast to interaction design and user experience design) implies more of a focus on assessing and making recommendations to improve usability than it does on design, though Usability Engineers may still engage in design to some extent, particularly design of wire-frames or other prototypes.

Standards and Guidelines

Usability engineers sometimes work to shape an interface such that it adheres to accepted operational definitions of user requirements. For example, the International Organisation for Standardisation-approved definitions usability are held by some to be a context-dependent yardstick for the effectiveness, efficiency and satisfaction with which specific users should be able to perform tasks. Advocates of this approach engage in task analysis, then prototype interface design, and usability testing on those designs. On the basis of such tests, the technology is (ideally) re-designed or (occasionally) the operational targets for user performance are revised. [Dillon, 2000].

The National Institute of Standards and Technology has collaborated with industry to develop the Common Industry Specification for Usability - Requirements which serves as a guide for many industry professionals. The specifications for successful usability in biometrics were also developed by the NIST. Usability.Gov provides a tutorial and wide general reference for the design of usable websites.

Usability, especially with the goal of Universal Usability, encompasses the standards and guidelines of design for accessibility. Some primary guidelines for web accessibility are:
1. The Web Accessibility Initiative Guidelines
2. The Section 508 government guidelines applicable to all public-sector websites.
3. The ADA Guidelines for accessibility of state and local government websites.
4. The IBM Guidelines for accessibility of websites.

Methods and Tools

Usability Engineers conduct usability evaluations of existing or proposed interfaces and their findings are fed back to the Designer for use in design or redesign. Common usability evaluation methods include:

- usability testing (Gold standard of Usability Engineering, but the most involved and expensive method)
- interviews
- focus groups
- questionnaires
- cognitive walkthroughs
- heuristic evaluations
- RITE method
- cognitive task analysis
- contextual inquiry
- Think aloud protocol

Usability testing, the gold standard, is when participants are recruited and asked to use the actual or prototype interface and their reactions, behaviors, errors, and self-reports in interviews are carefully observed and recorded by the Usability Engineer. On the basis of this data, the Usability Engineer recommends interface changes to improve usability.

There are a variety of online resources that make the job of the Usability Engineer a little easier. Some examples of these include:
1. The Web Metrics Tool Suite is a product of the National Institute of Standards and Technology. This toolkit is focused on evaluating the HTML of a website versus a wide range of usability guidelines and includes:

- Web Static Analyzer Tool (WebSAT) - checks web page HTML against typical usability guidelines
- Web Category Analysis Tool (WebCAT) - lets the usability engineer construct and conduct a web category analysis
- Web Variable Instrumenter Program (WebVIP) - instruments a website to capture a log of user interaction
- Framework for Logging Usability Data (FLUD) - a file format and parser for representation of user interaction logs
- FLUDViz Tool - produces a 2D visualization of a single user session
- VisVIP Tool - produces a 3D visualization of user navigation paths through a website
- TreeDec - adds navigation aids to the pages of a website

2. The Usability Testing Environment (UTE) produced by Mind Design Systems is available freely to federal government employees. According to the official company website this tool

"consists of two tightly-integrated applications. The first is the UTE Manager which helps a tester set up test scenarios (tasks) as well as survey and demographic questions. The UTE Manager also compiles the test results and produces customized reports and summary data which can be used as quantitative measures of usability observations and recommendations. The second UTE application is the UTE Runner. The UTE Runner presents the test participants with the test scenarios (tasks) as well as any demographic and survey questions. In addition, the UTE Runner tracks the actions of the subject throughout the test including clicks, keystrokes, and scrolling."

Research Resources

Some well-known practitioners in the field are Donald Norman, Jakob Nielsen, and John M. Carroll. Nielsen and Carroll have both written books on the subject of usability engineering. Nielsen's book is aptly titled *Usability Engineering*, and was published in 1993. Carroll wrote "Making Use: Scenario-Based Design of Human-Computer Interactions" in 2000, and co-authored "Usability Engineering: Scenario-Based Development of Human-Computer Interaction" with Mary Beth Rossen in 2001. Some other field leaders are Alan Cooper, Larry Constantine and Steve Krug the author of "*Don't Make Me Think! A Common Sense Approach to Web Usability*".

Chapter 9

Cognitive Walkthrough and Heuristic Evaluation

Cognitive walkthrough

The **cognitive walkthrough** method is a usability inspection method used to identify usability issues in a piece of software or web site, focusing on how easy it is for new users to accomplish tasks with the system. Whereas **cognitive walkthrough** is task-specific, heuristic evaluation takes a holistic view to catch problems not caught by this and other usability inspection methods. The method is rooted in the notion that users typically prefer to learn a system by using it to accomplish tasks, rather than, for example, studying a manual. The method is prized for its ability to generate results quickly with low cost, especially when compared to usability testing, as well as the ability to apply the method early in the design phases, before coding has even begun.

Introduction

A cognitive walkthrough starts with a task analysis that specifies the sequence of steps or actions required by a user to accomplish a task, and the system responses to those actions. The designers and developers of the software then walkthrough the steps as a group, asking themselves a set of questions at each step. Data is gathered during the walkthrough, and afterwards a report of potential issues is compiled. Finally the software is redesigned to address the issues identified.

The effectiveness of methods such as cognitive walkthroughs is hard to measure in applied settings, as there is very limited opportunity for controlled experiments while developing software. Typically measurements involve comparing the number of usability problems found by applying different methods. However, Gray and Salzman called into question the validity of those studies in their dramatic 1998 paper "Damaged Merchandise", demonstrating how very difficult it is to measure the effectiveness of usability inspection methods. However, the consensus in the usability community is that the cognitive walkthrough method works well in a variety of settings and applications.

Walking through the tasks

After the task analysis has been made the participants perform the walkthrough by asking themselves a set of questions for each subtask. Typically four questions are asked:

- **Will the user try to achieve the effect that the subtask has?** Does the user understand that this subtask is needed to reach the user's goal?
- **Will the user notice that the correct action is available?** E.g. is the button visible?
- **Will the user understand that the wanted subtask can be achieved by the action?** E.g. the right button is visible but the user does not understand the text and will therefore not click on it.
- **Does the user get feedback?** Will the user know that they have done the right thing after performing the action?

By answering the questions for each subtask usability problems will be noticed.

Common Mistakes

In teaching people to use the walkthrough method, Lewis & Rieman have found that there are two common misunderstandings :

1. The evaluator doesn't know how to perform the task themself, so they stumble through the interface trying to discover the correct sequence of actions -- and then they evaluate the stumbling process. (The user should identify and perform the **optimal** action sequence.)
2. The walkthrough does not test real users on the system. The walkthrough will often identify many more problems than you would find with a single, unique user in a single test session.

History

The method was developed in the early nineties by Wharton, et al., and reached a large usability audience when it was published as a chapter in Jakob Nielsen's seminal book on usability, "Usability Inspection Methods." The Wharton, et al. method required asking four questions at each step, along with extensive documentation of the analysis. In 2000 there was a resurgence in interest in the method in response to a CHI paper by Spencer who described modifications to the method to make it effective in a real software development setting. Spencer's streamlined method required asking only two questions at each step, and involved creating less documentation. Spencer's paper followed the example set by Rowley, et al. who described the modifications to the method that they made based on their experience applying the methods in their 1992 CHI paper "The Cognitive Jogthrough".

Heuristic evaluation

A **heuristic evaluation** is a discount usability inspection method for computer software that helps to identify usability problems in the user interface (UI) design. It specifically involves evaluators examining the interface and judging its compliance with recognized usability principles (the "heuristics"). These evaluation methods are now widely taught and practiced in the New Media sector, where UIs are often designed in a short space of time on a budget that may restrict the amount of money available to provide for other types of interface testing.

Introduction

The main goal of heuristic evaluations is to identify any problems associated with the design of user interfaces. Usability consultant Jakob Nielsen developed this method on the basis of several years of experience in teaching and consulting about usability engineering.

Heuristic evaluations are one of the most informal methods of usability inspection in the field of human-computer interaction. There are many sets of usability design heuristics; they are not mutually exclusive and cover many of the same aspects of user interface design.

Quite often, usability problems that are discovered are categorized—often on a numeric scale—according to their estimated impact on user performance or acceptance. Often the heuristic evaluation is conducted in the context of use cases (typical user tasks), to provide feedback to the developers on the extent to which the interface is likely to be compatible with the intended users' needs and preferences.

The simplicity of heuristic evaluation is beneficial at the early stages of design. This usability inspection method does not require user testing which can be burdensome due to the need for users, a place to test them and a payment for their time. Heuristic evaluation requires only one expert, reducing the complexity and expended time for evaluation. Most heuristic evaluations can be accomplished in a matter of days. The time required varies with the size of the artifact, its complexity, the purpose of the review, the nature of the usability issues that arise in the review, and the competence of the reviewers. Using heuristic evaluation prior to user testing will reduce the number and severity of design errors discovered by users. Although heuristic evaluation can uncover many major usability issues in a short period of time, a criticism that is often leveled is that results are highly influenced by the knowledge of the expert reviewer(s). This "one-sided" review repeatedly has different results than performance testing, each type of testing uncovering a different set of problems.

Nielsen's heuristics

Jakob Nielsen's heuristics are probably the most-used usability heuristics for user interface design. Nielsen developed the heuristics based on work together with Rolf Molich in 1990. The final set of heuristics that are still used today were released by Nielsen in 1994. The heuristics as published in Nielsen's book *Usability Engineering* are as follows

Visibility of system status:
The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

Match between system and the real world:
The system should speak the user's language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

User control and freedom:
Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

Consistency and standards:
Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

Error prevention:
Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

Recognition rather than recall:
Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

Flexibility and efficiency of use:
Accelerators—unseen by the novice user—may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

Aesthetic and minimalist design:
Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

Help users recognize, diagnose, and recover from errors:
Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

Help and documentation:
Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Gerhardt-Powals' cognitive engineering principles

Although Nielsen is considered the expert and field leader in heuristics, Jill Gerhardt-Powals also developed a set of cognitive principles for enhancing computer performance. These heuristics, or principles, are similar to Nielsen's heuristics but take a more holistic approach to evaluation. Gerhardt Powals' principles are listed below.

- **Automate unwanted workload**:
    - free cognitive resources for high-level tasks.
    - eliminate mental calculations, estimations, comparisons, and unnecessary thinking.

- **Reduce uncertainty**:
    - display data in a manner that is clear and obvious.

- **Fuse data**:
    - reduce cognitive load by bringing together lower level data into a higher-level summation.

- **Present new information with meaningful aids to interpretation**:
    - use a familiar framework, making it easier to absorb.
    - use everyday terms, metaphors, etc.

- **Use names that are conceptually related to function**:
    - Context-dependent.
    - Attempt to improve recall and recognition.
    - Group data in consistently meaningful ways to decrease search time.

- **Limit data-driven tasks**:
    - Reduce the time spent assimilating raw data.
    - Make appropriate use of color and graphics.

- **Include in the displays only that information needed by the user at a given time**.
- **Provide multiple coding of data when appropriate**.
- **Practice judicious redundancy**.

Chapter 10

User Interface Design

User interface design or **user interface engineering** is the design of computers, appliances, machines, mobile communication devices, software applications, and websites with the focus on the user's experience and interaction. The goal of user interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals—what is often called user-centered design. Good user interface design facilitates finishing the task at hand without drawing unnecessary attention to itself. Graphic design may be utilized to support its usability. The design process must balance technical functionality and visual elements (e.g., mental model) to create a system that is not only operational but also usable and adaptable to changing user needs.

Interface design is involved in a wide range of projects from computer systems, to cars, to commercial planes; all of these projects involve much of the same basic human interactions yet also require some unique skills and knowledge. As a result, designers tend to specialize in certain types of projects and have skills centered around their expertise, whether that be software design, user research, web design, or industrial design.

Processes

There are several phases and processes in the user interface design, some of which are more demanded upon than others, depending on the project.

- Functionality requirements gathering – assembling a list of the functionality required by the system to accomplish the goals of the project and the potential needs of the users.
- User analysis – analysis of the potential users of the system either through discussion with people who work with the users and/or the potential users themselves. Typical questions involve:
    - What would the user want the system to do?
    - How would the system fit in with the user's normal workflow or daily activities?
    - How technically savvy is the user and what similar systems does the user already use?
    - What interface look & feel styles appeal to the user?

- Information architecture – development of the process and/or information flow of the system (i.e. for phone tree systems, this would be an option tree flowchart and for web sites this would be a site flow that shows the hierarchy of the pages).
- Prototyping – development of wireframes, either in the form of paper prototypes or simple interactive screens. These prototypes are stripped of all look & feel elements and most content in order to concentrate on the interface.
- Usability testing – testing of the prototypes on an actual user—often using a technique called think aloud protocol where you ask the user to talk about their thoughts during the experience.
- Graphic Interface design – actual look & feel design of the final graphical user interface (GUI). It may be based on the findings developed during the usability testing if usability is unpredictable, or based on communication objectives and styles that would appeal to the user. In rare cases, the graphics may drive the prototyping, depending on the importance of visual form versus function. If the interface requires multiple skins, there may be multiple interface designs for one control panel, functional feature or widget. This phase is often a collaborative effort between a graphic designer and a user interface designer, or handled by one who is proficient in both disciplines.

User interface design requires a good understanding of user needs.

Requirements

The dynamic characteristics of a system are described in terms of dialogue requirements contained in seven principles of part 10 of the ergonomics standard, the ISO 9241. This standard establishes a framework of ergonomic "principles" for the dialogue techniques with high-level definitions and illustrative applications and examples of the principles. The principles of the dialogue represent the dynamic aspects of the interface and can be mostly regarded as the "feel" of the interface. The seven dialogue principles are:

- Suitability for the task: the dialogue is suitable for a task when it supports the user in the effective and efficient completion of the task.
- Self-descriptiveness: the dialogue is self-descriptive when each dialogue step is immediately comprehensible through feedback from the system or is explained to the user on request.
- Controllability: the dialogue is controllable when the user is able to initiate and control the direction and pace of the interaction until the point at which the goal has been met.
- Conformity with user expectations: the dialogue conforms with user expectations when it is consistent and corresponds to the user characteristics, such as task knowledge, education, experience, and to commonly accepted conventions.
- Error tolerance: the dialogue is error tolerant if despite evident errors in input, the intended result may be achieved with either no or minimal action by the user.
- Suitability for individualization: the dialogue is capable of individualization when the interface software can be modified to suit the task needs, individual preferences, and skills of the user.

- Suitability for learning: the dialogue is suitable for learning when it supports and guides the user in learning to use the system.

The concept of usability is defined in Part 11 of the ISO 9241 standard by effectiveness, efficiency, and satisfaction of the user. Part 11 gives the following definition of usability:

- Usability is measured by the extent to which the intended goals of use of the overall system are achieved (effectiveness).
- The resources that have to be expended to achieve the intended goals (efficiency).
- The extent to which the user finds the overall system acceptable (satisfaction).

Effectiveness, efficiency, and satisfaction can be seen as quality factors of usability. To evaluate these factors, they need to be decomposed into sub-factors, and finally, into usability measures.

The information presentation is described in Part 12 of the ISO 9241 standard for the organization of information (arrangement, alignment, grouping, labels, location), for the display of graphical objects, and for the coding of information (abbreviation, color, size, shape, visual cues) by seven attributes. The "attributes of presented information" represent the static aspects of the interface and can be generally regarded as the "look" of the interface. The attributes are detailed in the recommendations given in the standard. Each of the recommendations supports one or more of the seven attributes. The seven presentation attributes are:

- Clarity: the information content is conveyed quickly and accurately.
- Discriminability: the displayed information can be distinguished accurately.
- Conciseness: users are not overloaded with extraneous information.
- Consistency: a unique design, conformity with user’s expectation.
- Detectability: the user’s attention is directed towards information required.
- Legibility: information is easy to read.
- Comprehensibility: the meaning is clearly understandable, unambiguous, interpretable, and recognizable.

The user guidance in Part 13 of the ISO 9241 standard describes that the user guidance information should be readily distinguishable from other displayed information and should be specific for the current context of use. User guidance can be given by the following five means:

- Prompts indicating explicitly (specific prompts) or implicitly (generic prompts) that the system is available for input.
- Feedback informing about the user’s input timely, perceptible, and non-intrusive.
- Status information indicating the continuing state of the application, the system’s hardware and software components, and the user’s activities.
- Error management including error prevention, error correction, user support for error management, and error messages.

- On-line help for system-initiated and user initiated requests with specific information for the current context of use.

Research--Past and Ongoing

User interface design has been a topic of considerable research, including on its aesthetics. In the past standards have been developed, as far back as the eighties for defining the usablity of software products. One of the structural basis has become the IFIP userinterface reference model. The model proposes four dimensions to structure the user interface:

- The input/output dimension (the look)
- The dialogue dimension (the feel)
- The technical or functional dimension (the access to tools and services)
- The organizational dimension (the communication and co-operation support)

This model has greatly influenced the development of the international standard ISO 9241 describing the interface design requirements for usability. The desire to understand application-specific UI issues early in software development, even as an application was being developed, led to research on GUI rapid prototyping tools that might offer convincing simulations of how an actual application might behave in production use. Some of this research has shown that a wide variety of programming tasks for GUI-based software can, in fact, be specified through means other than writing program code.

Research in recent years is strongly motivated by the increasing variety of devices that can, by virtue of Moore's Law, host very complex interfaces.

There is also research on generating user interfaces automatically, to match a user's level of ability for different kinds of interaction.

Chapter 11

Card Sorting and Cognitive Dimensions of Notations

Card sorting

Card sorting is a simple technique in user experience design where a group of subject experts or "users", however inexperienced with design, are guided to generate a category tree or folksonomy. It is a useful approach for designing information architecture, workflows, menu structure, or web site navigation paths.

Card sorting has a characteristically low-tech approach. The concepts are first identified and written onto simple index cards or Post-it notes. The user group then arranges these to represent the groups or structures they are familiar with.

Groups may either be organised as collaborative groups (focus groups) or as repeated individual sorts. The literature discusses appropriate numbers of users needed to produce trustworthy results.

A card sort is commonly undertaken when designing a navigation structure for an environment that offers an interesting variety of content and functionality, such as a web site. In that context, the items to be organized are those that are significant in the environment. The way that the items are organized should make sense to the target audience and cannot be determined from first principles.

The field of information architecture is founded upon the study of the structure of information. If an accepted and standardized taxonomy exists for a subject, it would be natural to simply apply that taxonomy as a means of organizing both the information in the environment and any navigation to particular subjects or functions. Card sorting is applied when:

- The variety in the items to be organized is so great that no existing taxonomy is accepted as organizing the items.
- The similarities among the items make them difficult to divide clearly into categories.

- Members of the audience that uses the environment may differ significantly in how they view the similarities among items and the appropriate groupings of items.

Basic method

To perform a card sort:

1. A person representative of the audience is given a set of index cards with terms already written on them.
2. This person puts the terms into logical groupings, and finds a category name for each grouping.
3. This process is repeated across a population of test subjects.
4. The results are later analyzed to reveal patterns.

Open card sorting

In an **open card sort**, participants create their own names for the categories.

This helps reveal not only how they mentally classify the cards, but also what terms they use for the categories.

Open sorting is **generative**; it is typically used to discover patterns in how participants classify, which in turn helps generate ideas for organizing information.

Closed card sorting

In a **closed card sort**, participants are provided with a predetermined set of category names. They then assign the index cards to these fixed categories.

This helps reveal the degree to which the participants agree on which cards belong under each category.

Closed sorting is **evaluative**; it is typically used to judge whether a given set of category names provides an effective way to organize a given collection of content.

Reverse card sorting

In a *reverse card sort* or card-based classification, an existing structure of categories and sub-categories is tested. Users are given tasks and are asked to complete them navigating a collection of cards. Each card contains the names of subcategories related to a category, and the user should find the card most relevant to the given task starting from the main card with the top-level categories. This ensures that the structure is evaluated in isolation, nullifying the effects of navigational aids, visual design, and other factors.

Reverse card sorting is *evaluative*; it's used to judge whether a predetermined hierarchy provides a good way to find information.

Analyzing card-sort results

Various methods can be used to analyze the data. The purpose of the analysis is to extract patterns from the population of test subjects, so that a common set of categories and relationships emerges. This common set is then incorporated into the design of the environment, either for navigation or for other purposes.

Card sorting is an established technique with an emerging literature.

Online (remote) card sorting

There are a number of tools available to perform card sorting activities with survey participants via the internet. The perceived advantages of remote card sorting are that it allows a larger group of participants to be reached at a lower cost. The software can also assist in the process of analyzing card sort results. The advantages of a remote card sort must be traded off against the lack of personal interaction between card sort participants and the card sort administrator, which may produce valuable insights.

Cognitive dimensions of notations

Cognitive dimensions or **Cognitive dimensions of notations** are design principles for notations, user interfaces and programming language design, described by researchers Thomas R.G. Green and Marian Petre. The dimensions can be used to evaluate the usability of an existing *information artifact*, or as heuristics to guide the design of a new one.

Cognitive dimensions are designed to provide a lightweight approach to analysis of a design quality, rather than an in-depth, detailed description. They provide a common vocabulary for discussing many factors in notation, UI or programming language design. Also, cognitive dimensions help in exploring the space of possible designs through *design maneuvers*.

The authors identify four main user activities with interactive artifacts: *incrementation* [creation], *transcription*, *modification* and *exploratory design*. Each activity is best served by a different trade-off in the usability on each dimension. For example, a high viscosity (resistance to change) is harmful for modification and exploration activities, but less severe for the one-off tasks performed in transcription and incrementation.

Design manoeuvres

A **design maneuver** is a change in the notation design to alter its position within a particular dimension. Dimensions are created to be pairwise independent, so that the design can be altered in one dimension while keeping a second one constant.

But this usually results in a trade-off between dimensions. A modification increasing the usability of the notation in one dimension (while keeping a second one constant) will typically reduce its usability in a third dimension. This reflects an assumption in the framework that there is no perfect interface and that trade-offs are a fundamental part of usability design.

An example of a design maneuver is reducing the viscosity of a notation by adding abstraction mechanisms. Stylesheets are an example of a notation where this trade-off is made: a style sheet is an abstraction that represent the common styling attributes of items in a document. Changing the style sheet will modify all items at once, eliminating the *repetition viscosity* present in the need to change the style of each individual item.

List of the cognitive dimensions

Original set

Abstraction gradient

What are the minimum and maximum levels of abstraction exposed by the notation? Can details be encapsulated?

Closeness of mapping

How closely does the notation correspond to the problem world?

Consistency

After part of the notation has been learned, how much of the rest can be successfully guessed?

Diffuseness / terseness

How many symbols or how much space does the notation require to produce a certain result or express a meaning?

Error-proneness

To what extent does the notation influence the likelihood of the user making a mistake?

Hard mental operations

How much hard mental processing lies at the notational level, rather than at the semantic level? Are there places where the user needs to resort to fingers or penciled annotation to keep track of what's happening?

Hidden dependencies

Are dependencies between entities in the notation visible or hidden? Is every dependency indicated in both directions? Does a change in one area of the notation lead to unexpected side-effects?

Juxtaposability

Can different parts of the notation be compared side-by-side at the same time?

Premature commitment

Are there strong constraints on the order with which tasks must be accomplished? Are there decisions that must be made before all the necessary information is available? Can those decisions be reversed or corrected later?

Progressive evaluation

How easy is it to evaluate and obtain feedback on an incomplete solution?

Role-expressiveness

How obvious is the role of each component of the notation in the solution as a whole?

Secondary notation and escape from formalism

Can the notation carry extra information by means not related to syntax, such as layout, color, or other cues?

Viscosity

Are there in the notation any inherent barriers to change? How much effort is required to make a change to a program expressed in the notation?

This dimension can be further classified into the following types:

- 'Knock-On viscosity': a change in the code violates internal constraints in the program, whose resolution may violate further internal constraints.

- 'Repetition viscosity': a single action within the user's conceptual model requires many, repetitive device actions.
- 'Scope viscosity': a change in the size of the input data set requires changes to the program structure itself.

Visibility

How readily can required parts of the notation be identified, accessed and made visible?

New cognitive dimensions

Beyond the original set of cognitive dimensions defined by Thomas Green, new dimensions get proposed in the HCI research field with different levels of adoption and refinement. These candidate dimensions are based on research outside the Cognitive Dimensions framework, and are adapted to it as a way to summarize that research into the hands-on approach encouraged by it.

Some of these candidate dimensions include Creative ambiguity (does the notation encourage interpreting several meanings of the same element?), Indexing (are there elements to guide finding a specific part?), Synopsis ("gestalt view" of the whole annotated structure) or Unevenness (some creation paths are easier than others, which bias the expressed ideas in a developed artifact).

Chapter 12

Look and Feel & Partial Concurrent Thinking Aloud

Look and feel

In software design, ***look and feel*** is a term used in respect of a graphical user interface and comprises aspects of its design, including elements such as colors, shapes, layout, and typefaces (the "look"), as well as the behavior of dynamic elements such as buttons, boxes, and menus (the "feel"). The term can also refer to aspects of an API, mostly to parts of an API which are not related to its functional properties. The term is used in reference to both software and websites.

Look and feel applies to other products. In documentation, for example, it refers to the graphical layout (document size, color, font, etc.) and the writing style. In the context of equipment, it refers to consistency in controls and displays across a product line.

Look and feel in operating system user interfaces serves two general purposes. First, it provides branding, helping to identify a set of products from one company. Second, it increases ease of use, since users will become familiar with how one product functions (looks, reads, etc.) and can translate their experience to other products with the same look and feel.

In widget toolkits

Contrary to operating system user interfaces, for which look and feel is a part of the product identification, Widget toolkits often allow users to specialize their application look and feel, by deriving the default look and feel of the toolkit, or by completely defining their own. This specialization can go from skinning (which only deals with the look, or visual appearance of the widgets) to completely specializing the way the user interacts with the software (that is, the feel).

The definition of the look and feel to associate with the application is often done at initialization, but some Widget toolkits, such as the Swing widget toolkit that is part of the Java API, allow users to change the look and feel at runtime.

Some examples of Widget toolkits that support setting a specialized look and feel are:

- XUL (XML User Interface Language): The look and feel of the user interface can be specialized in a CSS file associated with the XUL definition files. Properties that can be specialized from the default are, for example, background or foreground colors of widgets, fonts, size of widgets, and so on.
- Swing supports specializing the look and feel of widgets by deriving from the default, another existing one, creating one from scratch, or beginning with J2SE 5.0, in an XML property file called synth (skinnable look and feel).

Lawsuits over

Some companies try to assert copyright of trade dress over their look and feel. Apple Computer was notable for its use of the term *look and feel* in reference to their Mac OS operating system. The firm tried, with some success, to block other software developers from creating software which had a similar look and feel. Apple argued that they had a copyright claim on the look and feel of their software, and even went so far as to sue Microsoft, alleging that the Windows operating system was illegally copying their look and feel.

Although provoking a vehement reaction in the software community, and causing Richard Stallman to form the League for Programming Freedom, the expected landmark ruling never happened, as most of the issues were resolved based on a license that Apple had granted Microsoft for Windows 1.0. *See: Apple v. Microsoft*. The First Circuit Court of Appeals rejected a copyright claim on the feel of a user interface in *Lotus v. Borland*.

Rainbow Arts' *Great Giana Sisters* was withdrawn after a lawsuit over the resemblance between its look and feel and that of *Super Mario Bros.*.

In APIs

An API, which is an interface to software which provides some sort of functionality, can also have a certain **look and feel**. Different parts of an API (e.g. different classes or packages) are often linked by common syntactic and semantic conventions (e.g. by the same asynchronous execution model, or by the same way object attributes are accessed). These elements are rendered either explicitly (i.e. are part of the syntax of the API), or implicitly (i.e. are part of the semantic of the API).

Partial concurrent thinking aloud

Partial Concurrent Thinking Aloud (or **partial concurrent think-aloud**, or **PCTA**) is a method used to gather data in usability testing with screen reader users. It is a particular kind of think aloud protocol (or TAP) created by Stefano Federici and Simone Borsci at the Interuniversity Center for Research on Cognitive Processing in Natural and Artificial Systems of University of Rome "La Sapienza". The Partial Concurrent Thinking Aloud is built up in order to create a specific usability assessment technique for blind users, eligible to maintain the advantages of concurrent and retrospective thinking aloud while overcoming their limits. Using PCTA blind users' verbalizations of problems could be more pertinent and comparable to those given by sighted people who use a concurrent protocol. In the usability evaluation with blind people, the retrospective thinking aloud is often adopted as a functional solution to overcome the structural interference due to thinking aloud and hearing the screen reader imposed by the classic thinking aloud technique; such a solution has yet a relapse in the evaluation method, because the concurrent and the retrospective protocols measure usability from different points of view, one mediated by navigation experience (retrospective) one more direct and pertinent (concurrent) . The use of PCTA could be widened to both summative and formative usability evaluations with mixed panels of users, thus extending the number of problems' verbalizations according to disabled users' divergent navigation processes and problem solving strategies.

Cognitive assumptions of Partial Concurrent Thinking Aloud

In general, in the usability evaluation both retrospective and concurrent TAP could be used according to the aims and goals of the study. Nevertheless, when a usability evaluation is carried out with blind people several studies propose to use the retrospective TAP: indeed, using a screen reader and talking about the way of interacting with the computer implie a structural interference between action and verbalization. Undoubtedly, cognitive studies provided a lot of evidence supporting the idea that individuals can listen, verbalize, or manipulate, and rescue information in multiple task condition. As Colin Cherry showed, subjects, when listening to two different messages from a single loudspeaker, can separate sounds from background noise, recognize the gender of the speaker, the direction, and the pitch (cocktail party effect). At the same time, subjects that must verbalize the content of a message (attended message) listening to two different message simultaneously (attended and unattended message) have a reduced ability to report the content of the attended massage, while they are unable to report the content of the unattended message. Moreover, K. Anders Ericsson and Walter Kintsch showed that, in a multiple task condition, subjects' ability of rescuing information is not compromised by an interruption of the action flow (as it happens in the concurrent thinking aloud technique), thanks to the "Long Term Working Memory mechanism" of information retrieval (Working Memory section Ericsson and Kintsch). Even if users can listen, recognize, and verbalize multiple messages in a multiple task condition and they can stop and restart actions without losing any information, other cognitive studies underlined that the overlap of activities in a multiple task condition have an effect on the goal achievement: Kemper, Herman and Lian , analysing the users' abilities to verbalize actions in a multiple task condition, showed that the fluency of a user’s conversation is influenced by the overlap of actions. Adults are likely to continue to talk as they navigate in a complex physical environment. However, the fluency of their conversation is likely to change: Older adults are likely to speak more slowly than they would if resting; Young adults continue to speak just as rapidly while walking as while resting, but they adopt a further set of speech accommodations, reducing sentence length, grammatical complexity, and propositional density. Just by reducing length, complexity, and propositional density adults free up working memory resources. We do not know how and how much the content of verbalizations could be influenced by the strategy of verbalization (i.e. the modification of fluency and the complexity in a multiple task condition). Anyway, we well know that users in the concurrent thinking aloud verbalize the problems in a more accurate and pertinent way (i.e. more focused on the problems directly perceived during the interaction) then in the retrospective one . The pertinence is granted to the user by the proximity of action-verbalization-next action; this multiple task proximity compels the subject to apply a strategy of verbalization that reduce the overload of the working memory. However, for blind users this time proximity between action and verbalization is lost: the use of the screen reader, in fact, increase the time for verbalization (i.e. in order to verbalize, blind users must first stop the [screen reader] and then restart it).

Protocol of Partial Concurrent Thinking Aloud

PCTA method is composed of two sections, one concurrent and one retrospective:

The first section is a modified concurrent protocol built up according to the three concurrent verbal protocols criteria described by K. Anders Ericsson and Herbert Simon:

The first criterion

Subjects should be talking about the task at hand, not about an unrelated issue. In order to respect this rule, the time between problem retrieval, thinking and verbalization must be minimized to avoid the influence of a long perceptual reworking and the consequent verbalization of unrelated issues. Blind participants, using a screen reader, increase the time latency between identification and verbalization of a problem. To minimize this latency, users are trained to ring a desk-bell that stops both time and navigation. During this suspension, users can create a memory sign (i.e. ring the bell) and restart immediately the navigation. This setting modification allows to avoid the cognitive limitation problem and the influence of perceptual reworking, also creating a memory sign for the retrospective analysis.

The second criterion

To be pertinent, verbalizations should be logically consistent with the verbalizations that just preceded them. For any kind of user it is hard to be pertinent and consistent in a concurrent verbal protocol. Therefore, the practitioners could generally interrupt the navigation and ask for a clarification or stimulate the users to verbalize in a pertinent way. In order to do so and stop navigation to screen reader users, we propose to negotiate a specific physical sign with them: The practitioner, sitting behind the user, will put his hand on the user's shoulder. This physical sign grants the verbalization pertinence and consistence.

The third criterion

A subset of the information needed during the task performance should be remembered. The concurrent model is based on the link between working memory and time latency. The proximity between the occurrence of a thought and its verbal report allows users to verbalize on the basis of their working memory.

The second PCTA section is a retrospective one in which users analyse those problems previously verbalized in a concurrent way. The memory signs, created by users ringing the desk-bell, overcome the limits of classic retrospective analysis; indeed, these signs allow the users to be pertinent and consistent with their concurrent verbalization, thus avoiding the influence of long term memory and perceptual reworking.

Chapter 13

Personalization

Personalization involves using technology to accommodate the differences between individuals. Once confined mainly to the Web, it is increasingly becoming a factor in education, health care (i.e. personalized medicine), television, and in both "business to business" and "business to consumer" settings.

Web pages

Web pages are personalized based on the characteristics (interests, social category, context, ...) of an individual. Personalization implies that the changes are based on implicit data, such as items purchased or pages viewed. The term *customization* is used instead when the site only uses explicit data such as ratings or preferences.

On an intranet or B2E Enterprise Web portals, **personalization** is often based on user attributes such as department, functional area, or role. The term **customization** in this context refers to the ability of users to modify the page layout or specify what content should be displayed.

There are three categories of personalizations:

1. Profile / Group based
2. Behaviour based
3. Collaboration based

Web personalization models include rules-based filtering, based on "if this, then that" rules processing, and collaborative filtering, which serves relevant material to customers by combining their own personal preferences with the preferences of like-minded others. Collaborative filtering works well for books, music, video, etc. However, it does not work well for a number of categories such as apparel, jewelry, cosmetics, etc. Recently, another method, "Prediction Based on Benefit", has been proposed for products with complex attributes such as apparel.

There are three broad methods of personalizations:

1. Implicit
2. Explicit

3. Hybrid

With implicit personalization the personalization is performed by the web page (or information system) based on the different categories mentioned above. With explicit personalization, the web page (or information system) is changed by the user using the features provided by the system. Hybrid personalization combines the above two approaches to leverage the *best of both worlds*.

Many companies offer services for web recommendation and email recommendation that are based on personalization or anonymously collected user behaviors. Following the example of Amazon.com, the online retailing industry has been early adopters of 3rd party personalization tools offered by companies like PredictiveIntent, MyBuys, Barilliance, iGoDigital and Certona. Other 3rd party vendors like Choice Stream or Vignette bring personalization to content and display advertising.

Web personalization is closely linked to the notion of **Adaptive hypermedia** (AH). The main difference is that the former would usually work on what is considered an Open Corpus Hypermedia, whilst the latter would traditionally work on Closed Corpus Hypermedia. However, recent research directions in the AH domain take both closed and open corpus into account. Thus, the two fields are closely inter-related.

Personalization is also being considered for use in less overtly commercial applications to improve the user experience online. Remote control manufacturer Ruwido developed an interactive IPTV platform in 2010 called Voco Media, which controls digital media in the living room using web personalization. It uses personalization as a tool that supports modern forms of TV usage, by allowing users to create different profiles for each family member, personalized menu structures and fingerprint recognition.

Digital media

Another aspect of personalization is the increasing prevalence of open data on the Web. Many companies make their data available on the Web via APIs, web services, and open data standards. Ordnance Survey Open Data This data is structured to allow it to be inter-connected and re-used by third parties.

Data available from a user's personal social graph can be accessed by third-party application software to be suited to fit the personalized web page or information appliance.

Current open data standards on the Web include:

1. DataPortability
2. OpenID
3. OpenSocial
4. Attention Profiling Mark-up Language (APML)

Mobile Phones

Over time mobile phones have seen an increased emphasis placed on user personalization. Far from the black and white screens and monophonic ringtones of the past, phones now offer interactive wallpapers and MP3 TruTones. In the UK and Asia, WeeMees have become popular. WeeMees are three-dimensional characters that are used as wallpaper and respond to the tendencies of the user. Video Graphics Array (VGA) picture quality allows people to change their background with ease without sacrificing quality. All of these services are downloaded through the provider with the goal to make the user feel connected to the phone.

Television

Personalization on the TV can occur on TV apps or on the set top box user interface. Most forms of personalization occur with recommendations. For instance, a TV app may recommend certain TV shows based on user behaviour or collaborative filtering.

Search engines

When search engines employ personalization, it is referred to as personalized search. Google was the first of the major Web search engines to introduce personalized results on a massive scale. Weighing a number of factors including but not limited to user history, bookmarks, community behaviour and site click-through rate and stickiness, Google is providing results that are specific to what they believe you are searching for. Currently this service is only available to those who are logged into their Google account. Bing personalizes search results for all users based on an individual's previous searches.

Amazon.com has been the early adopter of personalization technology to recommend products to shoppers on its site, based upon their previous purchases. Amazon makes extensive use of Collaborative Filtering in its personalization technology.

TV Genius has developed personalization in its television search engine for TV show searches.

Print media

In print media, ranging from magazines to promotional publications, personalization uses databases of individual recipients' information. Not only does the written document address itself by name to the reader, but the advertising is targeted to the recipient's demographics or interests using fields within the database, such as "first name", "last name", "company", etc.

The term "personalization" should not be confused with variable data, which is a much more granular method of marketing that leverages both images and text with the medium, not just fields within a database. Although personalized children's books are created by companies who are using and leveraging all the strengths of Variable Data Printing. This allows for full image and text variability within a printed book.

Promotional merchandise

Promotional items industry (mugs, T-shirts, keychains, balls etc.) are regularly personalized. Personalized children's storybooks — wherein the child becomes the protagonist, with the name and image of the child personalized — are also popular. Personalized CDs for children also exist. With the advent of digital printing, personalized calendars that start in any month, Birthday Cards, cards, e-cards, posters and Photo Book can also be obtained.

Mass personalization

Mass personalization is defined as custom tailoring by a company in accordance with its end users tastes and preferences. The main difference between mass customization and mass personalization is that customization is the ability for a company to give its customers an opportunity to create and choose product to certain specifications, but does have limits.

Behavioral targeting represents a concept that is similar to mass personalization.

Advantages of personalization and mass customization

Sphere Sovereignty and Discretionary Control: Users like to have the sphere sovereignty and elements of discretionary control that personalization provides.

Save time: Eliminate repetitive tasks; remember transactional details; recognize habits and shorten the path to engage in such habits (example: frequently called numbers on a phone should automatically go into the phone's memory).

Save money: Prevent redundant work (example: make it easier for employees and suppliers to know someone else has already solved the problem that they are currently facing); eliminate service components unnecessary to a customer; identify lower-cost solutions that meet all other specifications.

Better information: Provide training; filter out information not relevant to a person; provide more specific information that is increasingly relevant to a person's interests; increase the reliability of information; replace "average" information with information specific to that person's environment.

Address ongoing needs, challenges, or opportunities: Provide one-stop services; allow flexibility in work hours, job responsibilities, and benefits; accommodate unique personal preferences (example: allow employees to customize their office space, within certain boundaries); recognize and reward achievement with special treatment.

Personalization allows a company to tailor a specific product in accordance with individual standards, tastes and preferences. For example, baseball jerseys can be customized based on size, colour, team and logo, however there are a finite number of choices for these variables to choose. To personalize a jersey, a name or number can be administered to it as well as custom fitting.

The emergence of e-commerce has allowed for the personalization of clothing as well as the customization of audio CD's and downloading of music as well as graphic design for personal websites from the comfort of one's own home. Computer companies have been widely regarded as a market leader in made to order desktops or notebooks for high-performance and entertainment needs. Consumers are able to place orders based on product family, usage, price range, processor, and form factors. This customization ensures that each purchaser can view the merchandise available in order to make an informed decision.

Disadvantages to personalization

No matter how remarkable or laudable a company's efforts at personalization, there will always be some people who simply are not interested. Every firm must be prepared to recognize and instantly accommodate any of the motivating factors that would cause a person to decide he or she doesn't want any sort of personalization.

Technological barriers mean that it can be difficult to accurately deploy personalization solutions.

For example, the technology to easily type and enter information on a TV is limited. Most remote controllers can only interact with the TV on a basic level, which makes filling out any kind of personal profile tedious. However, this can be over come with personalization on the mobile or web applications, or by developing implicit profiles.

From an individual's perspective, there are numerous situations or attitudes that make personalization unwelcome.

Anonymity preferred. There are many reasons why people might not want to be identified, from the innocent - it's a birthday present they don't want their spouse to discover in advance on their credit card statement - to the unethical or illegal. Some people are simply private, and prefer to mind their own business and let others mind theirs. Others recognize the growing infringements on private space and choose to take the cautious route. A. Michael Froomkin, associate professor at the University of Miami School of Law, wrote, "Anonymity may be the primary tool available to citizens to combat the compilation and analysis of personal profile data, although data protection laws also may have some effect."

Lack of relevance. People do not want a relationship with companies that have no relevance to them. Computer programmers have no interest in getting to know an executive recruiter who only places sales executives. Homeowners who only buy the finest products for their home will not be interested in a cut-rate furniture store. If you've never been to Arkansas, never plan to go there, and don't know anyone there, you don't want to be on the mailing list of the Arkansas Tourism Board. On the Web, companies constantly ignore this factor and ask individuals for information before demonstrating to the person's satisfaction that their services are relevant. The prime example is companies that insist people fill out a lengthy form before they can gain access to a demo or to additional information. If a company asks people for information before it has demonstrated relevance, between 30 and 50 percent-depending on which statistics you believe-will lie to prevent revealing personal information.

Lack of credibility. If you don't trust a company, it becomes a relationship of last resort. Unless you have no choice, you don't want to deal with it. People don't need proof that a company deserves to be in this category. Often, a small suggestion that this might be the case is enough to justify caution.

Lack of security. Good intentions aren't enough. If a company fails to protect its assets, and those of its stakeholders, then people will not be willing to share anything of value with the firm. Security is like sausage making: the more you know about it, the less likely you are to be comfortable. People have real reasons to fear that today's centralized networks are not secure, because they frequently are not.

Technology firms are working to solve security problems, although most admit that security is a process, not a single technological solution. There are no quick fixes.

Impossible. Sometimes, people just aren't able to take advantage of attractive offers. If a company, local government, spouse, or neighborhood forbids a person from moving forward, that's life. Likewise, if people lack the ability to accept personalization-perhaps they lack a sophisticated enough cell phone, or a fast enough Web connection-it won't happen.

Infrequent contact. People will have little interest in establishing a relationship with a cab driver in a city they rarely visit, or with the company that installs their new septic system (a once-in-twenty-five-years event.) Companies get around this limitation by broadening their services to increase the frequency of contacts. Hewlett-Packard's printer division used to focus on selling printers; now the firm realizes it can make more money selling printer cartridges, as well as paper, and in the process increase the frequency of its interactions with customers.

Little value placed on potential benefits. People may not recognize the value in offered personalization, such as when firms offer to customize product offers. Many people don't want to receive any such offers, period. Employees who are offered personalized training may not value it if they were unimpressed with their previous experiences with the training unit, and thus believe that even personalization won't make the time invested worthwhile.

As companies move towards the personal - and the number of interactions increases - it's important to gain greater objectivity about the attractiveness of a firm's offers. Today, in the early stages of our shift towards increasingly personal business relationships, most personalization is still superficial, and way too much of it is mainly personalized marketing. No matter how targeted advertising becomes, it still won't be anything more than a means to an end, and too much of it is flat-out annoying. People tolerate occasional annoyances, but when annoyances multiply, they begin to reek of harassment.

Even highly attractive offers won't make a difference to a person who doesn't value the potential benefits. Think about a new knowledge management system that theoretically delivers "better" information by filtering out "less relevant" citations. Many researchers may cringe at the thought, because they succeed by looking at raw data and thus understanding at a deeper level the background and related elements of a given situation.

Chapter 14

Progressive Enhancement

Progressive enhancement is a strategy for web design that emphasizes accessibility, semantic HTML markup, and external stylesheet and scripting technologies. Progressive enhancement uses web technologies in a layered fashion that allows everyone to access the basic content and functionality of a web page, using any browser or Internet connection, while also providing those with better bandwidth, more advanced browser software or more experience an enhanced version of the page.

History

"Progressive Enhancement" was coined by Steven Champeon of hesketh.com in a series of articles and presentations for Webmonkey and the SXSW Interactive conference between March and June 2003.

Introduction and background

The strategy is an attempt to subvert the traditional web design strategy known as "graceful degradation", wherein designers would create Web pages for the latest browsers that would also work well in older versions of browser software. Graceful degradation was supposed to allow the page to "degrade", or remain presentable even if certain technologies assumed by the design were not present, without being jarring to the user of such older software. In practice, graceful degradation has been supplanted by an attitude that the end user should "just upgrade". This attitude is due to time and budget constraints, limited access to testing alternate browser software, as well as the widespread belief that "browsers are free". Unfortunately, upgrading is often not possible due to IT department policies, older hardware, and other reasons. The "just upgrade" attitude also ignores deliberate user choices and the existence of a variety of browser platforms; many of which run on handhelds or in other contexts where available bandwidth is restricted, or where support for sound or color and limited screen size, for instance, are far different from the typical graphical desktop browser. However there is also a large group of internet users unaware or uncaring about the features of the browser.

In Progressive Enhancement (PE) the strategy is deliberately reversed: a basic markup document is created, geared towards the lowest common denominator of browser software functionality, and then the designer adds in functionality or enhancements to the presentation and behavior of the page, using modern technologies such as Cascading Style Sheets or JavaScript (or other advanced technologies, such as Flash or Java applets or SVG, etc.) All such enhancements are externally linked, preventing data unusable by certain browsers from being unnecessarily downloaded.

The PE approach is derived from Champeon's early experience (c. 1993-4) with SGML, before working with HTML or any Web presentation languages, as well as from later experiences working with CSS to work around browser bugs. In those early SGML contexts, semantic markup was of key importance, whereas presentation was nearly always considered separately, rather than being embedded in the markup itself. This concept is variously referred to in markup circles as the rule of *separation of presentation and content*, *separation of content and style*, or of *separation of semantics and presentation*. As the Web evolved in the mid-nineties, but before CSS was introduced and widely supported, this cardinal rule of SGML was repeatedly violated by HTML's extenders. As a result, web designers were forced to adopt new, disruptive technologies and tags in order to remain relevant. With a nod to graceful degradation, in recognition that not everyone had the latest browser, many began to simply adopt design practices and technologies only supported in the most recent and perhaps the single previous major browser releases. For several years, much of the Web simply did not work in anything but the most recent, most popular browsers. This remained true until the rise and widespread adoption of and support for CSS, as well as many populist, grassroots educational efforts (from Eric Costello, Owen Briggs, Dave Shea, and others) showing Web designers how to use CSS for layout purposes.

PE is based on a recognition that the core assumption behind "graceful degradation" — that browsers always got faster and more powerful — was proving itself false with the rise of handheld and PDA devices with low-functionality browsers and serious bandwidth constraints. In addition, the rapid evolution of HTML and related technologies in the early days of the Web has slowed, and very old browsers have become obsolete, freeing designers to use powerful technologies such as CSS to manage all presentation tasks and JavaScript to enhance complex client-side behavior.

First proposed as a somewhat less unwieldy catchall phrase to describe the delicate art of "separating document structure and contents from semantics, presentation, and behavior", and based on the then-common use of CSS hacks to work around rendering bugs in specific browsers, the PE strategy has taken on a life of its own as new designers have embraced the idea and extended and revised the approach.

Core principles

Progressive Enhancement consists of the following core principles:

- basic content should be accessible to all browsers
- basic functionality should be accessible to all browsers
- sparse, semantic markup contains all content
- enhanced layout is provided by externally linked CSS

- enhanced behavior is provided by unobtrusive, externally linked JavaScript
- end user browser preferences are respected

Support and adoption

- Jim Wilkinson created a page for Progressive Enhancement to collect some tricks and tips and to explain the overall strategy.
- Designers such as Jeremy Keith have shown how the approach can be used harmoniously with still other approaches to modern Web design (such as Ajax) to provide flexible, but powerful, user experiences.
- Others, including Dave Shea, have helped to spread the adoption of the term to refer to CSS-based design strategies.
- Organizations such as the Web Standards Project have embraced PE as a basis for their educational efforts.
- In 2006 Nate Koechley at Yahoo! made extensive reference to PE in his own approach to Web design and browser support, Graded Browser Support (GBS).
- Steve Chipman at AOL has referred to PE as a basis for his Web design strategy.
- David Artz, leader of the AOL Optimization team, developed a suite of Accessible Rendering Technologies, and invented a technique for disassembly of the “enhancement” on the fly, saving the user's preference.
- Chris Heilmann discusses the importance of targeted delivery of CSS so that each browser only gets the content (and enhancements) it can handle.
- Scott Jehl of Filament Group introduces "Test-Driven Progressive Enhancement" on A List Apart. The article proposes testing browser capabilities (rather than user-agent detection) before applying enhancements, to ensure a usable experience to the largest audience possible.
- Wt is a web application library which transparently implements progressive enhancement during its bootstrap, progressing from plain HTML to full AJAX.
- EnhanceJS is a Javascript framework maintained by Filament Group designed to improve the application of Progressive Enhancement by first testing browser capabilities for key Javascript and CSS support before applying advanced styles and scripts to the page.

Benefits for accessibility

Web pages created according to the principles of PE are by their nature more accessible, because the strategy demands that basic content always be available, not obstructed by commonly unsupported or easily disabled scripting. Additionally, the sparse markup principle makes it easier for tools that read content aloud to find that content. It is unclear as to how well PE sites work with older tools designed to deal with table layouts, "tag soup," and the like.

Benefits for search engine optimization (SEO)

Improved results with respect to Search Engine Optimization is another side effect of a PE-based Web design strategy. Because the basic content is always accessible to search engine spiders, pages built with PE methods avoid problems that may hinder search engine indexing.

Criticism and responses

Some skeptics, such as Garret Dimon, have expressed their concern that PE is not workable in situations that rely heavily on JavaScript to achieve certain user interface presentations or behaviors. Others have countered with the point that informational pages should be coded using PE in order to be indexed by spiders, and that even Flash-heavy pages should be coded using PE. In a related area, many have expressed their doubts concerning the principle of the separation of content and presentation in absolute terms, pushing instead for a realistic recognition that the two are inextricably linked.

Chapter 15

Seven Stages of Action

"Seven Stages of Action" is a term coined by the usability consultant Donald Norman. He explains this phrase in chapter two of his book *The Design of Everyday Things* in the context of explaining the psychology of a person behind the task performed by him or her.

Building up the Stages

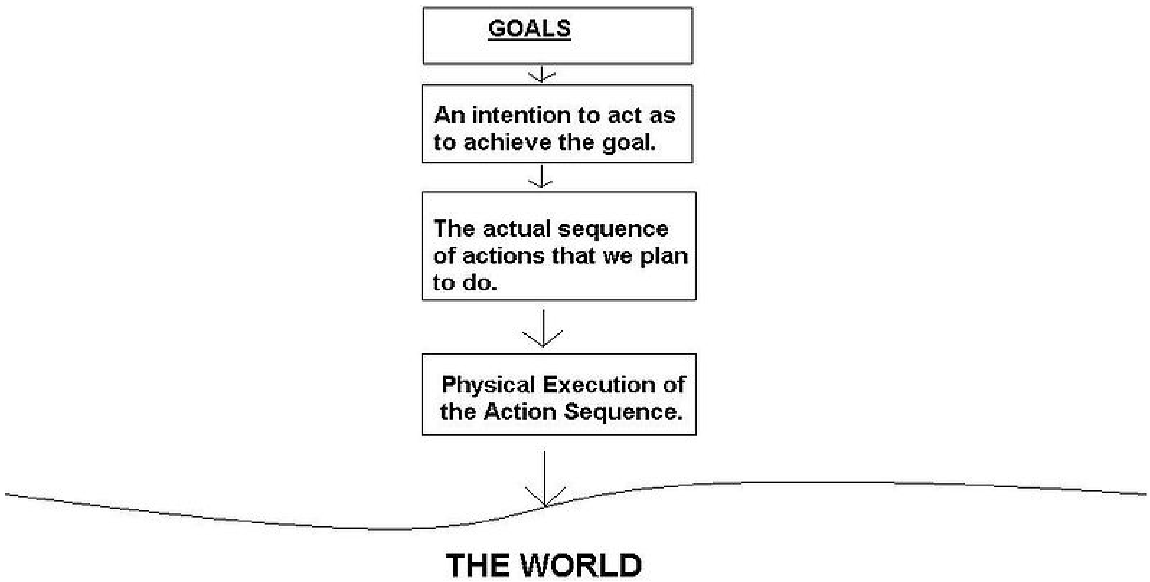
History

The history behind the **Action Cycle** (inset:picture - right) starts from a conference in Italy attended by Donald Norman. This excerpt has been taken from the book "Design of Everyday Things":-

"I am in Italy at a conference. i watch the next speaker attempt to thread a film onto a projector that he never used before. He puts the reel into place, then takes it off and reverses it. Another person comes to help. Jointly they thread the film through the projector and hold the free end, discussing how to put it on the takeup reel. Two more people come over to help and then another. The voices grow louder, in three languages: Italian, German and English. One person investigates the controls, manipulating each and announcing the result. Confusion mounts. I can no longer observe all that is happening. The conference organizer comes over. After a few moments he turns and faces he audience, which has been waiting patiently in the auditorium. "Ahem," he says, "is anybody expert in projectors?" Finally, fourteen minutes after the speaker had started to thread the film (and eight minutes after the scheduled start of the session) a blue-coated technician appears. He scowls, then promptly takes the entire film off the projector, rethreads it, and gets it working."

Norman pondered on the reasons that made something like threading of a projector difficult to do. To examine this, he wanted to know what happened when something implied nothing. In order to do that, he examined the structure of an action. So to get something done, a notion of what is wanted - the goal that is to be achieved, needs to be started. Then, something is done to the world i.e. take action to move oneself or manipulate someone or something. Finally, the checking is required if the goal was made. This led to formulation of Stages of Execution and Evaluation.

Stages of Execution



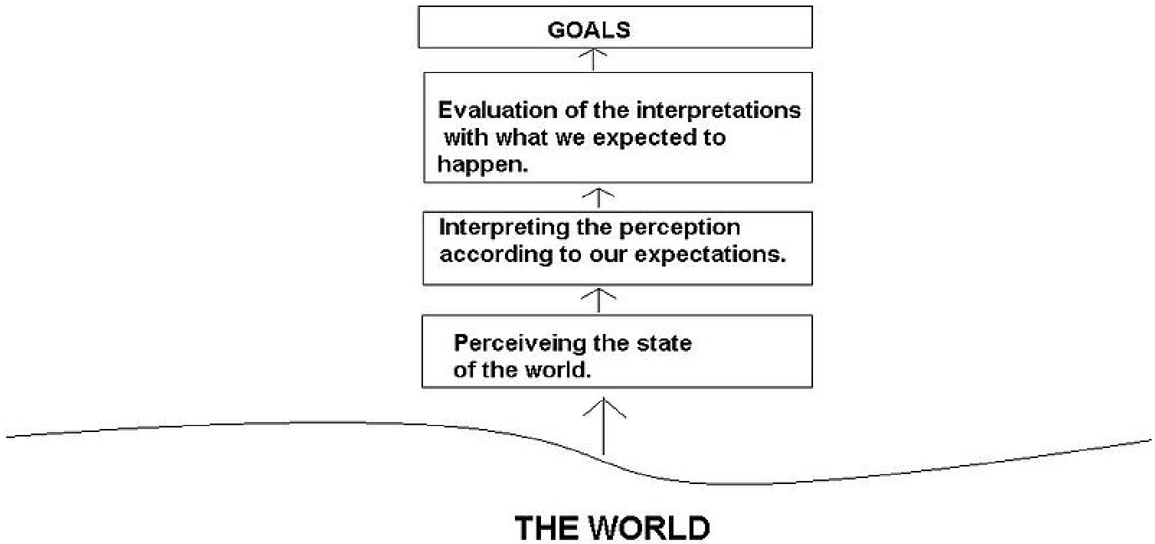
Stages of Execution as described by Donald Norman.

Execution formally means to perform or do something. Norman explains that a person sitting on an armchair while reading a book during dusk time, might need more light when it becomes dimmer and dimmer. To do that, he needs to switch on the button of a lamp i.e. get more light (the goal). To do this, one must need to specify on how to move one's body, how to stretch to reach the light switch and how to extend one's finger to push the button. The goal has to be translated into an intention, which in turn has to be made into an action sequence.

Thus, formulation of stages of execution:-

- Start at the top with the *goal*, the state that is to be achieved.
- The goal is translated into an *intention* to do some action.
- The intention must be translated into a set of internal commands, an *action sequence* that can performed to satisfy the intention.
- The action sequence is still a mutual even: nothing happens until it is *executed*, performed upon the world.

Stages of Evaluation



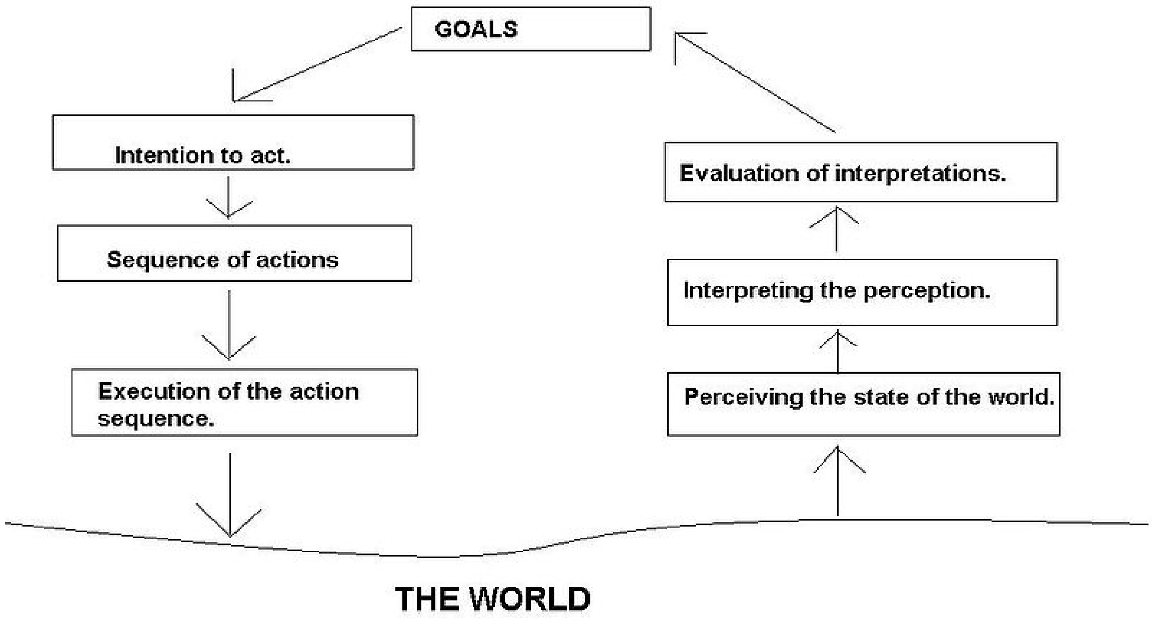
Stages of Evaluation as described by Donald Norman.

Evaluation formally means to examine and calculate. Norman explains that after turning on the light, we evaluate if it is actually turned on. A careful judgement is then passed on how the light has affected our world i.e. the room in which the person is sitting on the armchair while reading a book.

The formulation of the stages of evaluation can be described as:-

- Evaluation starts with our *perception* of the world.
- This perception must then be *interpreted* according to our expectations.
- Then it is compared *(evaluated)* with respect to both our intentions and our goals.

Seven Stages of Action



Seven Stages of Action as described by Donald Norman.

Seven Stages of Action constitute three stages of execution, three stages of evaluation and our goals.

1. Forming the goal

2. Forming the intention

3. Specifying an action

4. Executing the action

5. Perceiving the state of the world

6. Interpreting the state of the world

7. Evaluating the outcome

The Gulfs

The Gulf of Execution

The difference between the intentions and the allowable actions is the *Gulf of execution*.

"Consider the movie projector example: one problem resulted from the Gulf of Execution. The person wanted to set up the projector. Ideally, this would be a simple thing to do. But no, a long, complex sequence was required. It wasn't all clear what actions had to be done to accomplish the intentions of setting up the projector and showing the film."

The Gulf of Evaluation

The *Gulf of evaluation* reflects the amount of effort that the person must exert to interpret the physical state of the system and to determine how well the expectations and intentions have been met.

"In the movie projector example there was also a problem with the Gulf of Evaluation. Even when the film was in the projector, it was difficult to tell if it had been threaded correctly."

Usage as Design Aids

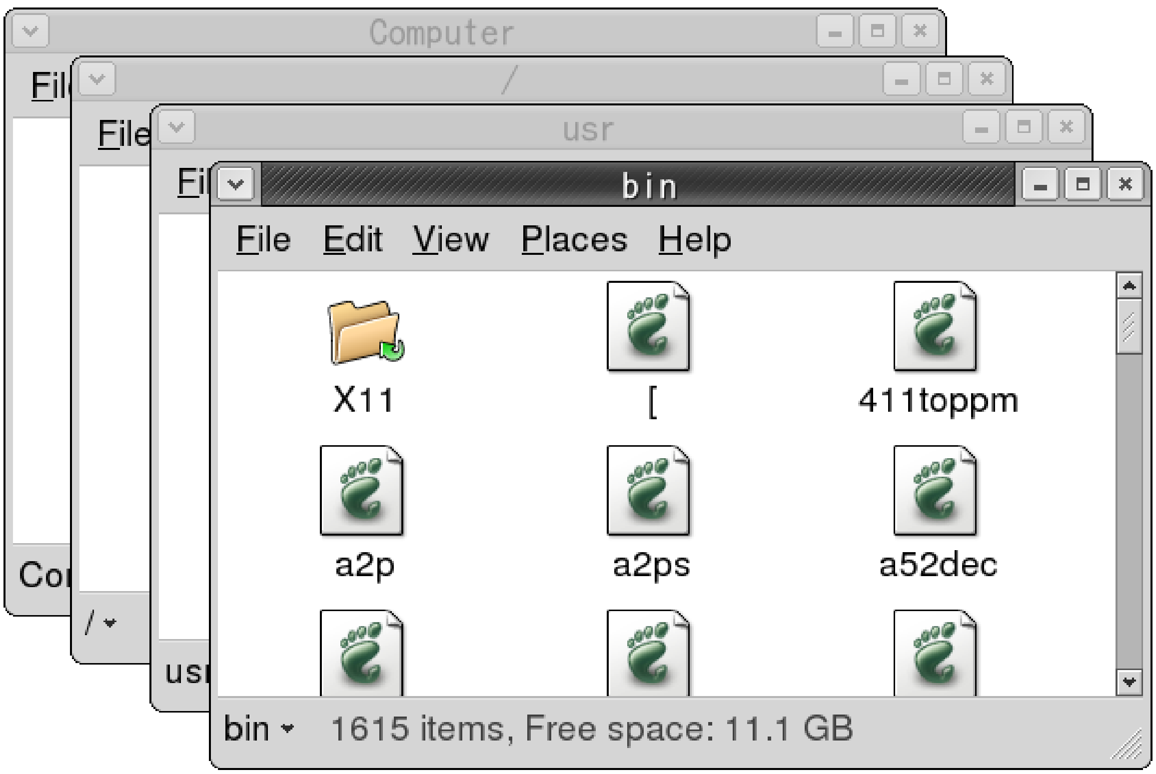
The seven-stage structure is referenced as design aid to act as a basic checklist for designers' questions to ensure that the Gulfs of Execution and Evaluation are bridged.

The Seven Stages of Action can be broken down into 4 main principles of good design:-

- *Visibility* - By looking, the user can tell the state of the device and the alternatives for action.
- *A Good Conceptual Model* - The designer provides a good conceptual model for the user, with consistency in the presentation of operations and results and a coherent, consistent system image.
- *Good mappings* - It is possible to determine the relationships between actions and results, between the controls and their effects, and between the system state and what is visible.
- *Feedback* - The user receives full and continuous feedback about the results of the actions.

Chapter 16

Spatial File Manager



Nautilus is a file manager with a spatial mode. Each of these windows is an open folder.

In computing, a **spatial file manager** is a file manager that uses a spatial metaphor to represent files and folders as if they are real physical objects.

Concepts

The base requirements of a spatial file manager are:

1. Each folder is represented by a single window.
2. Each window is unambiguously and irrevocably tied to a particular folder.

3. Stability: files, folders, and windows go where the user moves them, stay where the user puts them ("preserve their spatial state"), and retain all their other "physical" characteristics (like size, shape, color and location).
4. The same item can only be viewed in one window at a time.

Given an arbitrary window in a spatial file manager, it must be possible to determine with complete certainty which folder that window represents. Furthermore, it must not be possible to change that association.

Traditionally, when a folder is opened, the icon representing the folder changes—perhaps from an image showing a closed drawer to an opened one, perhaps the folder's icon turns into a silhouette filled with a pattern—and a new window is opened. Attempting to open that "already opened" folder will simply reveal the existing window. A new window will not be created because that would violate requirement number one listed above, attempting to reveal an already opened folder's contents using another window will either close the existing folder before opening it or refuse to reveal said contents. The change in the folder icon's state is meant to be a visual reminder of this behavior. It says, "This folder is already open." Similarly, while a document or application is open, their icons also represent this status and re-opening them will just reveal and bring them back to the front.

A common alternative to the spatial file manager is the navigational file manager or "browser-style" file manager. A window in such a system is a device through which the contents of many folders may be viewed. A browser-style window is not tied to any particular folder. The window's spatial state (size, position, etc.) stays the same as the contents of many different folders are viewed through it. More than one browser window may show the contents of a particular folder simultaneously.

Examples

Examples of file managers that use a spatial metaphor by default include:

- Apple's Finder 5 to 9 (versions up to Mac OS X)
- the RISC OS Filer
- the ROX Desktop's ROX-Filer
- Amiga's Workbench
- the OS/2 Workplace Shell
- GNOME's Nautilus from version 2.6 till 2.29
- BeOS's Tracker
- Windows Explorer in Windows 95 was also a spatial file manager in some respects, but became a navigational file manager in later versions of the operating system (when using the default settings)
- Enlightenment 17's file manager

Other objects

Some file managers represent other objects, such as a trash can for unwanted files, or computer or floppy disk icons to represent storage media. There may also be special folders that contain special items such as user preferences or the user's applications.

Advantages and disadvantages

Advantages

A spatial file manager is closer to the way the people interact with the physical objects around them, such as desks, chairs and tables. Since spatial properties are reliable indicators of "location" in a spatial file manager, the user may build up a familiar work environment by arranging commonly used windows, secure in the knowledge that this state will be preserved and will always map to the expected locations. This form of "nesting" (in the avian sense, not the hierarchical one) can lead to increased comfort and efficiency.

Furthermore, identification based on spatial attributes is a very natural human ability, requiring little or no conscious thought. The ability to recognize and recall locations within the hierarchy based on the appearance and position of folder windows is the primary purpose of the spatial file manager. All of the "rules" and behaviors that define the spatial file manager are designed to ensure that the strengths of the visual/spatial recognition and recall abilities of the human brain are leveraged. The idea is that these abilities are more natural and require "less work" than other forms of recognition based on reading text, maintaining an awareness of "current working directory" (in a command-line environment, for example), relying on the memory of past actions, or any other non-spatial cues.

One more advantage is that it allows the user to keep things arranged a certain way from one session to the next, as with the "workspace" settings in many high-end software packages. For instance, the windows showing the contents of different folders for a complex project could be tiled onscreen in such a fashion that all of them could be seen at the same time. In a browser-style file manager, one would be forced to rearrange these same windows every time they were reopened.

Disadvantages

The spatial metaphor can seem awkward to those accustomed to browser-style file managers. One complaint is that spatial managers use too many windows, leading to clutter. In a Windows 95 usability study by Microsoft, users of varying experience were said to be "confused" by the many windows:

"Users of every type were confused by the Programs folder. We thought that having a folder on the desktop with other folders and links to programs inside it would be a natural transition for Windows 3.1 users accustomed to Program Manager, while being relatively easy to learn for beginners. We were wrong! Beginners quickly got lost in all of the folders (unlike File Cabinet, each folder opened into a different window) and other users had a lot of trouble deciding whether they were looking at the actual file system and its files or just links to actual files."

Proponents claim that this confusion is partially a result of the non-spatial nature of the Windows 95 file manager. When the connection between the spatial state of a window is not unambiguously and irrevocably connected with a particular folder, it becomes impossible to reliably recognize a particular folder based on its spatial qualities. Spatial state often becomes misleading in a non-spatial file manager. Users may incorrectly identify a window based on visual cues that do not actually link it to any particular folder, but are instead properties of the browser-style window itself. In such an environment, each new window adds clutter without recognizable meaning. This leads to the often-cited preference for a single window through which any folder may be viewed: a browser. The proliferation and familiarity of web browsers has strengthened this preference.

Also, maintaining spatial familiarity can be difficult when the file system is accessed from a variety of applications and devices with differing display capabilities. Reproducing a single spatial arrangement on many different display devices is sometimes impractical. Per-device, per-user, or per-display spatial state is one possible solution, at the cost of an increased amount of state information that must be stored.

Chapter 17

User-Centered Design

In broad terms, **user-centered design (UCD)** or **pervasive usability** is a design philosophy and a process in which the needs, wants, and limitations of end users of a product are given extensive attention at each stage of the design process. User-centered design can be characterized as a multi-stage problem solving process that not only requires designers to analyze and foresee how users are likely to use a product, but also to test the validity of their assumptions with regards to user behaviour in real world tests with actual users. Such testing is necessary as it is often very difficult for the designers of a product to understand intuitively what a first-time user of their design experiences, and what each user's learning curve may look like.

The chief difference from other product design philosophies is that user-centered design tries to optimize the product around how users can, want, or need to use the product, rather than forcing the users to change their behavior to accommodate the product.

UCD models and approaches

For example, the user-centered design process can help software designers to fulfill the goal of a product engineered for their users. User requirements are considered right from the beginning and included into the whole product cycle. These requirements are noted and refined through investigative methods including: ethnographic study, contextual inquiry, prototype testing, usability testing and other methods. Generative methods may also be used including: card sorting, affinity diagraming and participatory design sessions. In addition, user requirements can be inferred by careful analysis of usable products similar to the product being designed.

- Cooperative design: involving designers and users on an equal footing. This is the Scandinavian tradition of design of IT artifacts and it has been evolving since 1970.
- Participatory design (PD), a North American term for the same concept, inspired by Cooperative Design, focusing on the participation of users. Since 1990, there has been a bi-annual Participatory Design Conference.
- Contextual design, “customer-centered design” in the actual context, including some ideas from Participatory design

All these approaches follow the ISO standard Human-centred design for interactive systems (ISO 9241-210, 2010).

Purpose

UCD answers questions about users and their tasks and goals, then uses the findings to make decisions about development and design. UCD of a web site, for instance, seeks to answer the following questions:

- Who are the users of the document?
- What are the users' tasks and goals?
- What are the users' experience levels with the document, and documents like it?
- What functions do the users need from the document?
- What information might the users need, and in what form do they need it?
- How do users think the document should work?

Elements

As examples of UCD viewpoints, the essential elements of UCD of a web site are considerations of visibility, accessibility, legibility and language.

Visibility

Visibility helps the user construct a mental model of the document. Models help the user predict the effect(s) of their actions while using the document. Important elements (such as those that aid navigation) should be emphatic. Users should be able to tell from a glance what they can and cannot do with the document.

Accessibility

Users should be able to find information quickly and easily throughout the document, regardless of its length. Users should be offered various ways to find information (such as navigational elements, search functions, table of contents, clearly labeled sections, page numbers, color coding, etc). Navigational elements should be consistent with the genre of the document. 'Chunking' is a useful strategy that involves breaking information into small pieces that can be organized into some type meaningful order or hierarchy. The ability to skim the document allows users to find their piece of information by scanning rather than reading. Bold and italic words are often used.

Legibility

Text should be easy to read: Through analysis of the rhetorical situation, the designer should be able to determine a useful font style. Ornamental fonts and text in all capital letters are hard to read, but italics and bolding can be helpful when used correctly. Large or small body text is also hard to read. (Screen size of 10-12 pixel sans serif and 12-16 pixel serif is recommended.) High figure-ground contrast between text and background increases legibility. Dark text against a light background is most legible.

Language

Depending on the rhetorical situation, certain types of language are needed. Short sentences are helpful, as well as short, well-written texts used in explanations and similar bulk-text situations. Unless the situation calls for it, do not use jargon or technical terms. Many writers will choose to use active voice, verbs (instead of noun strings or nominals), and simple sentence structure.

Rhetorical situation

A user-centered design is focused around the rhetorical situation. The rhetorical situation shapes the design of an information medium. There are three elements to consider in a rhetorical situation: Audience, Purpose, and Context.

Audience

The audience is the people who will be using the document. The designer must consider their age, geographical location, ethnicity, gender, education, etc.

Context

The context is the circumstances surrounding the situation. The context often answers the question: What situation has prompted the need for this document? Context also includes any social or cultural issues that may surround the situation.

Analysis Tools Used in User Centered Design

There are a couple of main tools that are used in the analysis of user centered design, mainly: persona, scenarios, and essential use cases.

Persona

During the UCD process, a persona of the user's need may be created. It is a fictional character with all the characteristics of the user. Personas are created after the field research process, which typically consists of members of the primary stakeholder (user) group answering questionnaires or participating in interviews, or a mixture of both. After results are gathered from the field research, they are used to create personas of the primary stakeholder group. Often, there may be several personas concerning the same group of individuals, since it is almost impossible to apply all the characteristics of the stakeholder group onto one character. The character depicts the "typical" or "average" individual in the primary stakeholder group, and is referred to throughout the entire design process. There are also what's called a secondary persona, where the character is not a member of the primary stakeholder group and is not the main target of the design, but their needs should be met and problems solved if possible. They exist to help account for further possible problems and difficulties that may occur even though the primary stakeholder group is satisfied with their solution. There is also an anti-persona, which is the character which the design process is not made for. Personas usually include a name and picture, demographics, roles and responsibilities, goals and tasks, motivations and needs, environment and context, and a quote that can represent the character's personality. Personas are useful in the sense that they create a common shared understanding of the user group for which the design process is built around. Also, they help to prioritize the design considerations by providing a context of what the user needs and what functions are simply nice to add and have. They can also provide a human face and existence to a diversified and scattered user group, and can also create some empathy and add emotions when referring to the users. However, since personas are a generalized perception of the primary stakeholder group from collected data, the characteristics may be too broad and typical, or too much of an "average joe". Sometimes, personas can have stereotypical properties also, which may hurt the entire design process. Overall, personas are a useful tool that can be used since designers in the design process can have an actual person to make design measure around other than referring to a set of data or a wide range of individuals.

Scenario

A scenario created in the UCD process is a fictional story about the "daily life of" or a sequence of events with the primary stakeholder group as the main character. Typically, a persona that was created earlier is used as the main character of this story. The story should be specific of the events happening that relate to the problems of the primary stakeholder group, and normally the main research questions the design process is built upon. These may turn out to be a simple story about the daily life of an individual, but small details from the events should imply details about the users, and may include emotional or physical characteristics. There can be the "best case scenario", where everything works out best for the main character, the "worst case scenario", where the main character experiences everything going wrong around him or her, and an "average case scenario", which is the typical life of the individual, where nothing really special or really depressing occurs, and the day just moves on. Scenarios create a social context to which the personas exist in, and also create an actual physical world, instead of imagining a character with internal characteristics from gathered data an nothing else; there is more action involved in the persona's existence. A scenario is also more easily understood by people, since it is in the form of a story, and is easier to follow. Yet, like the personas, these scenarios are assumptions made by the researcher and designer, and is also created from a set of organized data. Some even say such scenarios are unrealistic to real life occurrences. Also, it is difficult to explain and inform low level tasks that occur, like the thought process of the persona before acting.

Use Case

In short, a use case describes the interaction between an individual and the rest of the world. Each use case describes an event that may occur for a short period of time in real life, but may consist of intricate details and interactions between the actor and the world. It is represented as a series of simple steps for the character to achieve his or her goal, in the form of a cause-and effect scheme. Use cases are normally written in the form of a chart with two columns: first column labelled actor, second column labelled world, and the actions performed by each side written in order in the respective columns. The following is an example of a use case for performing a song on a guitar in front of an audience.

Actor	World
choose music to play	
pick up guitar	
	display sheet music
perform each note on sheet music using guitar	
	convey note to audience using sound
	audience provides feedback to performer
assess performance and adjust as needed based on audience feedback	
complete song with required adjustments	
	audience applause

The interaction between actor and the world is an act that can be seen in everyday life, and we take them as granted and don't think too much about the small detail that needs to happen in order for an act like performing a piece of music to exist. It is similar to the fact that when speaking our mother tongue, we don't think too much about grammar and how to phrase words; they just come out since we are so used to saying them. The actions between an actor and the world, notably, the primary stakeholder (user) and the world in this case, should be thought about in detail, and hence use cases should are created to take these tiny interactions to occur.

An essential use case is a special kind of use case, also called an "abstract use case". Essential use cases describe the essence of the problem, and deals with the nature of the problem itself. While writing use cases, no assumptions about unrelated details should be made. In additions, the goals of the subject should be separated from the process and implementation to reach that particular goal. Below is an example of an essential use case with the same goal as the former example.

Actor	World
choose sheet music to perform	
gathers necessary resources	
	provides access to resources
performs piece sequentially	

convey and interprets performance
provides feedback

completes performance

Use cases are useful because they help identify useful levels of design work. They allow the designers to see the actual low level processes that are involved for a certain problem, which makes the problem easier to handle, since certain minor steps and details the user makes are exposed. The designers' job should take into consideration of these small problems in order to arrive at a final solution that works. Another way to say this is that use cases breaks a complicated task in to smaller bits, where these bits are useful units. Each bit completes a small task, which then builds up to the final bigger task. Like writing code on a computer, it is easier to write the basic smaller parts and make them work first, and then put them together to finish the larger more complicated code, instead to tackling the entire code from the very beginning. The first solution is less risky because if something goes wrong with the code, it is easier to look for the problem in the smaller bits, since the segment with the problem will be the one that does not work, while in the latter solution, the programmer may have to look through the entire code to search for a single error, which proves time consuming. The same reasoning goes for writing use cases in UCD. Lastly, use cases convey useful and important tasks where the designer can see which one are of higher importance than others. Some drawbacks of writing use cases include the fact that each actions, by the actor or the world, consist of little detail, and is simply a small action. This may possibly lead to further imagination and different interpretation of action from different designers. Also, during the process, it is really easy to over-simplify a task, since a small task from a larger task may consist of even smaller tasks. Picking up a guitar may involved thinking of which guitar to pick up, which pick to use, and think about where the guitar is located first. These tasks may then be divided into smaller tasks, such as first thinking of what colour of guitar fits the place to perform the piece, and other related details. Tasks may be split further down into even tinier tasks, and it is up to the designer to determine what is a suitable place to stop splitting up the tasks. Tasks may not only be oversimplified, they may also be omitted in whole, thus the designer should be aware of all the detail and all the key steps that are involved in an event or action when writing use cases.

User-centered design, needs and emotions

The book "The Design of Everyday Things" (originally called "The Psychology of Everyday Things") was first published in 1986. In this book, Donald A. Norman describes the psychology behind what he deems 'good' and 'bad' design through examples and offers principles of 'good' design. He exalts the importance of design in our everyday lives, and the consequences of errors caused by bad designs.

In his book, Norman uses the term "user-centered design" to describe design based on the needs of the user, leaving aside what he considers secondary issues like aesthetics. User-centered design involves simplifying the structure of tasks, making things visible, getting the mapping right, exploiting the powers of constraint, and designing for error. Norman's overly reductive approach in this text was readdressed by him later in his own publication "Emotional Design."

Other books in a similar vein include "Designing Pleasurable Products" by Patrick W. Jordan, in which the author suggests that different forms of pleasure should be included in a user-centered approach in addition to traditional definitions of usability.

User-centered design in product lifecycle management systems

Software applications (or often suites of applications) used in product lifecycle management (typically including CAD, CAM and CAx processes) can be typically characterized by the need for these solutions to serve the needs of a broad range of users, with each user having a particular job role and skill level. For example, a CAD digital mockup might be utilized by a a novice analyst, design engineer of moderate skills, or a manufacturing planner of advanced skills.

Chapter 18

Web Accessibility

Web accessibility refers to the inclusive practice of making websites usable by people of all abilities and disabilities. When sites are correctly designed, developed and edited, all users can have equal access to information and functionality. For example, when a site is coded with semantically meaningful HTML, with textual equivalents provided for images and with links named meaningfully, this helps blind users using text-to-speech software and/or text-to-Braille hardware. When text and images are large and/or enlargable, it is easier for users with poor sight to read and understand the content. When links are underlined (or otherwise differentiated) as well as coloured, this ensures that color blind users will be able to notice them. When clickable links and areas are large, this helps users who cannot control a mouse with precision. When pages are coded so that users can navigate by means of the keyboard alone, or a single switch access device alone, this helps users who cannot use a mouse or even a standard keyboard. When videos are closed captioned or a sign language version is available, deaf and hard of hearing users can understand the video. When flashing effects are avoided or made optional, users prone to seizures caused by these effects are not put at risk. And when content is written in plain language and illustrated with instructional diagrams and animations, users with dyslexia and learning difficulties are better able to understand the content. When sites are correctly built and maintained, all of these users can be accommodated while not impacting on the usability of the site for non-disabled users.

The needs that Web accessibility aims to address include:

- **Visual:** Visual impairments including blindness, various common types of low vision and poor eyesight, various types of color blindness;
- **Motor/Mobility:** e.g. difficulty or inability to use the hands, including tremors, muscle slowness, loss of fine muscle control, etc., due to conditions such as Parkinson's Disease, muscular dystrophy, cerebral palsy, stroke;
- **Auditory:** Deafness or hearing impairments, including individuals who are hard of hearing;
- **Seizures:** Photoepileptic seizures caused by visual strobe or flashing effects.
- **Cognitive/Intellectual:** Developmental disabilities, learning disabilities (dyslexia, dyscalculia, etc.), and cognitive disabilities of various origins, affecting memory, attention, developmental "maturity," problem-solving and logic skills, etc.

Assistive technologies used for web browsing

Individuals living with a disability use assistive technologies such as the following to enable and assist web browsing:

- Screen reader software, which can read out, using synthesized speech, either selected elements of what is being displayed on the monitor (helpful for users with reading or learning difficulties), or which can read out everything that is happening on the computer (used by blind and vision impaired users).
- Braille terminals, consisting of a Refreshable Braille display which renders text as Braille characters (usually by means of raising pegs through holes in a flat surface) and either a QWERTY or Braille keyboard.
- Screen magnification software, which enlarges what is displayed on the computer monitor, making it easier to read for vision impaired users.
- Speech recognition software that can accept spoken commands to the computer, or turn dictation into grammatically correct text - useful for those who have difficulty using a mouse or a keyboard.
- Keyboard overlays, which can make typing easier and more accurate for those who have motor control difficulties.

Guidelines on accessible web design

Web Content Accessibility Guidelines

In 1999 the Web Accessibility Initiative, a project by the World Wide Web Consortium (W3C), published the Web Content Accessibility Guidelines WCAG 1.0. In recent years, these have been widely accepted as the definitive guidelines on how to create accessible websites.

On 11 December 2008, the WAI released the WCAG 2.0 as a Recommendation. WCAG 2.0 aims to be up to date and more technology neutral.

Criticism of WAI guidelines

For a general criticism of the W3C process, read Putting the user at the heart of the W3C process. There was a formal objection to WCAG's original claim that WCAG 2.0 will address requirements for people with learning disabilities and cognitive limitations headed by Lisa Seeman and signed by 40 organisations and people. In articles such as WCAG 2.0: The new W3C guidelines evaluated, To Hell with WCAG 2.0 and Testability Costs Too Much, the WAI has been criticised for allowing WCAG 1.0 to get increasingly out of step with today's technologies and techniques for creating and consuming web content, for the slow pace of development of WCAG 2.0, for making the new guidelines difficult to navigate and understand, and other argued failings.

Other guidelines

Canada

Canada has the Common Look and Feel Standards requiring federal government internet websites to meet Web Content Accessibility Guidelines (WCAG) 1.0 Checkpoints Priorities 1 and 2 (Double A conformance level). The standards have existed since 2000 and were updated in 2007.

Philippines

As part of the Web Accessibility Initiatives in the Philippines, the government through the National Council for the Welfare of Disabled Persons (NCWDP) board approved the recommendation of forming an adhoc or core group of webmasters that will help in the implementation of the Biwako Millennium Framework set by the UNESCAP.

The Philippines was also the place where the Interregional Seminar and Regional Demonstration Workshop on Accessible Information and Communications Technologies (ICT) to Persons with Disabilities was held where eleven countries from Asia - Pacific were represented. The Manila Accessible Information and Communications Technologies Design Recommendations was drafted and adopted in 2003.

Spain

In Spain, UNE 139803 is the norm entrusted to regulate web accessibility. This standard is based on Web Content Accessibility Guidelines 1.0.

Sweden

In Sweden, Verva, the Swedish Administrative Development Agency is responsible for a set of guidelines for Swedish public sector web sites. Through the guidelines, Web accessibility is presented as an integral part of the overall development process and not as a separate issue.

The Swedish guidelines contain criteria which cover the entire lifecycle of a website; from its conception to the publication of live web content. These criteria address several areas which should be considered, including:

- accessibility
- usability
- web standards
- privacy issues
- information architecture
- developing content for the web
- Content Management Systems (CMS) / authoring tools selection.
- development of web content for mobile devices.

An English translation was released in April 2008: Swedish National Guidelines for Public Sector Websites

The translation is based on the latest version of Guidelines which was released in 2006.

United Kingdom

In December 2010, the BSI (British Standards Institute) released the standard BS 8878:2010 Web accessibility. Code of practice. This standard effectively supersedes PAS 78 (pub. 2006). PAS 78, produced by the The Disability Rights Commission and British Standards Institution, provided guidance to organisations in how to go about commissioning an accessible website from a design agency. It describes what is expected from websites to comply with the UK Disability Discrimination Act 1995 (DDA), making websites accessible to and usable by disabled people.

BS 8878:2010 Web accessibility - Code of Practice. The standard has been designed to introduce non-technical professionals to improved accessibility, usability and user experience for disabled and older people. It will be especially beneficial to anyone new to this subject as it gives guidance on process, rather than on technical and design issues. BS 8878 is consistent with the Equality Act 2010 and is referenced in the UK government's e-Accessibility Action Plan as the basis of updated advice on developing accessible online services. It includes recommendations for:

- Involving disabled people in the development process and using automated tools to assist with accessibility testing
- The management of the guidance and process for upholding existing accessibility guidelines and specifications.

BS 8878 is intended for anyone responsible for the policies covering web product creation within their organization, and governance against those policies (e.g. Chief Executive Officers, Managing Directors, Headteachers, ICT managers). It would also assist:

- People responsible for promoting and supporting equality and inclusion initiatives within an organization (e.g. Human Resource (HR) managers or those responsible for Corporate Social Responsibility - CSR).
- Procurement managers (e.g. those responsible for procuring web products or the tools to create them such as content production systems or virtual learning environments).
- Web production teams (e.g. product owners, project managers, technical architects and web developers, designers, usability and accessibility engineers, test engineers).
- People with responsibility for creating or shaping online content (e.g. website editors, marketing managers, web content authors).
- People who create web production, testing or validation tools.

- People who write and deliver training courses in web production, design or coding.

Other audiences that might also be interested in this British Standard include:

- Assistive technology creators, vendors and trainers who need insights into how their technologies impact on the production of accessible web products.
- Those disabled and older people whose web accessibility needs the Standard aims to support and present.

Japan

Web Content Accessibility Guidelines in Japan was established in 2004 as JIS (Japanese Industrial Standards) X 8341-3. JIS X 8341-3 will be revised within 2009 by adopting WCAG 2.0. New version will have the same 4 principles, 12 guidelines, and 61 success criteria as WCAG 2.0 has.

Essential Components of Web Accessibility

In order for the web to be accessible, 7 components must be included:

1. the content on Web pages must be natural information (text, images, and sound),
2. Web browsers and media players,
3. assistive technologies,
4. users' knowledge and experience using the Web,
5. developers,
6. authoring tools
7. evaluation tools

These components interact with each other to create an environment that is accessible to people with disabilities.

Web **developers** usually use **authoring tools** and evaluation tools to create Web **content**. **People** ("**users**") use Web **browsers**, **media players**, **assistive technologies** or other "**user agents**" to get and interact with the **content**."

Guidelines for Different Components

Authoring Tool Accessibility Guidelines (ATAG)

- ATAG contains 28 checkpoints that provide guidance on:
    - producing accessible output that meets standards and guidelines
    - promoting the content author for accessibility-related information
    - providing ways of checking and correcting inaccessible content
    - integrating accessibility in the overall look and feel
    - making the authoring tool itself accessible to people with disabilities

Web Content Accessibility Guidelines (WCAG)

- WCAG 1.0: 14 guidelines that are general principles of accessible design
- WCAG 2.0: 12 principal guidelines

User Agent Accessibility Guidelines (UAAG)

- UAAG contains a comprehensive set of checkpoints that cover:
    - access to all content
    - user control over how content is rendered
    - user control over the user interface
    - standard programming interfaces

Legally required web accessibility

A growing number of countries around the world have introduced legislation which either directly addresses the need for websites and other forms of communication to be accessible to people with disabilities, or which addresses the more general requirement for people with disabilities not to be discriminated against.

Australia

In 2000, an Australian blind man won a court case against the Sydney Organizing Committee of the Olympic Games (SOCOG). This was the first successful case under Disability Discrimination Act 1992 because SOCOG had failed to make their official website, Sydney Olympic Games, adequately accessible to blind users. The Human Rights and Equal Opportunity Commission (HREOC) also published World Wide Web Access: Disability Discrimination Act Advisory Notes. All Governments in Australia also have policies and guidelines that require accessible public websites; Vision Australia maintain a complete list of Australian web accessibility policies.

Ireland

In Ireland, the Disability Act 2005 was supplemented with the National Disability Authority's Code of Practice on Accessible Public Services in July 2006. It is a practical guide to help all Government Departments and nearly 500 public bodies to comply with their obligations under the Disability Act 2005.

United Kingdom

In the UK, the Disability Discrimination Act 1995 (DDA) does not refer explicitly to website accessibility, but makes it illegal to discriminate against people with disabilities. The DDA applies to anyone providing a service; public, private and voluntary sectors. The Code of Practice: Rights of Access - Goods, Facilities, Services and Premises document published by the government's Disability Rights Commission to accompany the Act does refer explicitly to websites as one of the "services to the public" which should be considered covered by the Act.

Website accessibility audits

A growing number of organizations, companies and consultants offer *website accessibility audits*. These audits, a type of system testing, identify accessibility problems that exist within a website, and provide advice and guidance on the steps that need to be taken to correct these problems.

A range of methods are used to audit websites for accessibility:

- Automated tools are available which can identify some of the problems that are present.
- Expert technical reviewers, knowledgeable in web design technologies and accessibility, can review a representative selection of pages and provide detailed feedback and advice based on their findings.
- User testing, usually overseen by technical experts, involves setting tasks for ordinary users to carry out on the website, and reviewing the problems these users encounter as they try to carry out the tasks.

Each of these methods has its strengths and weaknesses:

- Automated tools can process many pages in a relatively short length of time, but can only identify some of the accessibility problems that might be present in the website.
- Technical expert review will identify many of the problems that exist, but the process is time consuming, and many websites are too large to make it possible for a person to review every page.
- User testing combines elements of usability and accessibility testing, and is valuable for identifying problems that might otherwise be overlooked, but needs to be used knowledgeably to avoid the risk of basing design decisions on one user's preferences.

Ideally, a combination of methods should be used to assess the accessibility of a website.

Accessible Web applications and WAI-ARIA

For a Web page to be accessible all important semantics about the page's functionality must be available so that assistive technology can understand and process the content and adapt it for the user. However as content becomes more and more complex, the standard HTML tags and attributes become inadequate in providing semantic reliably. Modern Web applications often apply scripts to elements to control their functionality and to enable them to act as a control or other dynamic component. These custom components or widgets do not provide a way to convey semantic information to the user agent. WAI-ARIA (Accessible Rich Internet Applications) is a specification published by the World

Wide Web Consortium that specifies how to increase the accessibility of dynamic content and user interface components developed with Ajax, HTML, JavaScript and related technologies. ARIA enables accessibility by enabling the author to provide all the semantics to fully describe its supported behaviour. It also allows each element can expose its current states and properties and its relationships between other elements. Accessibility problems with the focus and tab index are also corrected.