HUMAN COMPUTER INTERACTION

An Introduction to Human Computer Interaction

Dr. Debasis Samanta

INDIAN INSTITUTE OF TECHNOLOGY, KHARAGPUR

1.1. Definition of "Human Computer Interaction"

Human Computer Interaction is the study of interaction between users and computers. There is currently no agreed upon definition of the range of topics which form the area of human-computer interaction. Yet we need a characterization of the field if we are to derive and develop educational materials for it. Therefore a working definition has been offered that at least permits us to get down to the practical work of deciding what is to be taught.

Definition according to ACM SIGCHI*: Human-computer interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them. (Reference. 1)*

Regardless of the definition chosen, HCI is clearly to be included as a part of computer science and is as much a part of computer science as it is a part of any other discipline. If, for example, one adopts Newell, Perlis, and Simon's (1967) classic definition of computer science as "the study of computers and the major phenomena that surround them," then the interaction of people and computers and the uses of computers are certainly parts of those phenomena. If, on the other hand, we take the recent ACM (Denning, et al., 1988) report's definition as "the systematic study of algorithmic processes that describe and transform information: their theory, analysis, design, efficiency, implementation, and application," then those algorithmic processes clearly include interaction with users just as they include interaction with other computers over networks. The algorithms of computer graphics, for example, are just those algorithms that give certain experiences to the perceptual apparatus of the human. The design of many modern computer applications inescapably requires the design of some component of the system that interacts with a user. Moreover, this component typically represents more than half a system's lines of code. It is intrinsically necessary to understand how to decide on the functionality a system will have, how to bring this out to the user, how to build the system, how to test the design.

Because human-computer interaction studies a human and a machine in communication, 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 performance are relevant. And, of course, engineering and design methods are relevant.

1.2 HCI – A Multiplinary Discipline

HCI draws attention from different fields. Apart from Computer Science, Electronics and IT, it draws attention from several other fields like Cognitive and behavioral science, Human factors, some empirical studies, Interface device development, Graphical design and lots more.

The fields have been discussed while concerning the features of Human Computer Interaction. According to ACM SIGCHI, Computer Science in the Basic discipline and other disciplines serves as supportive discipline.

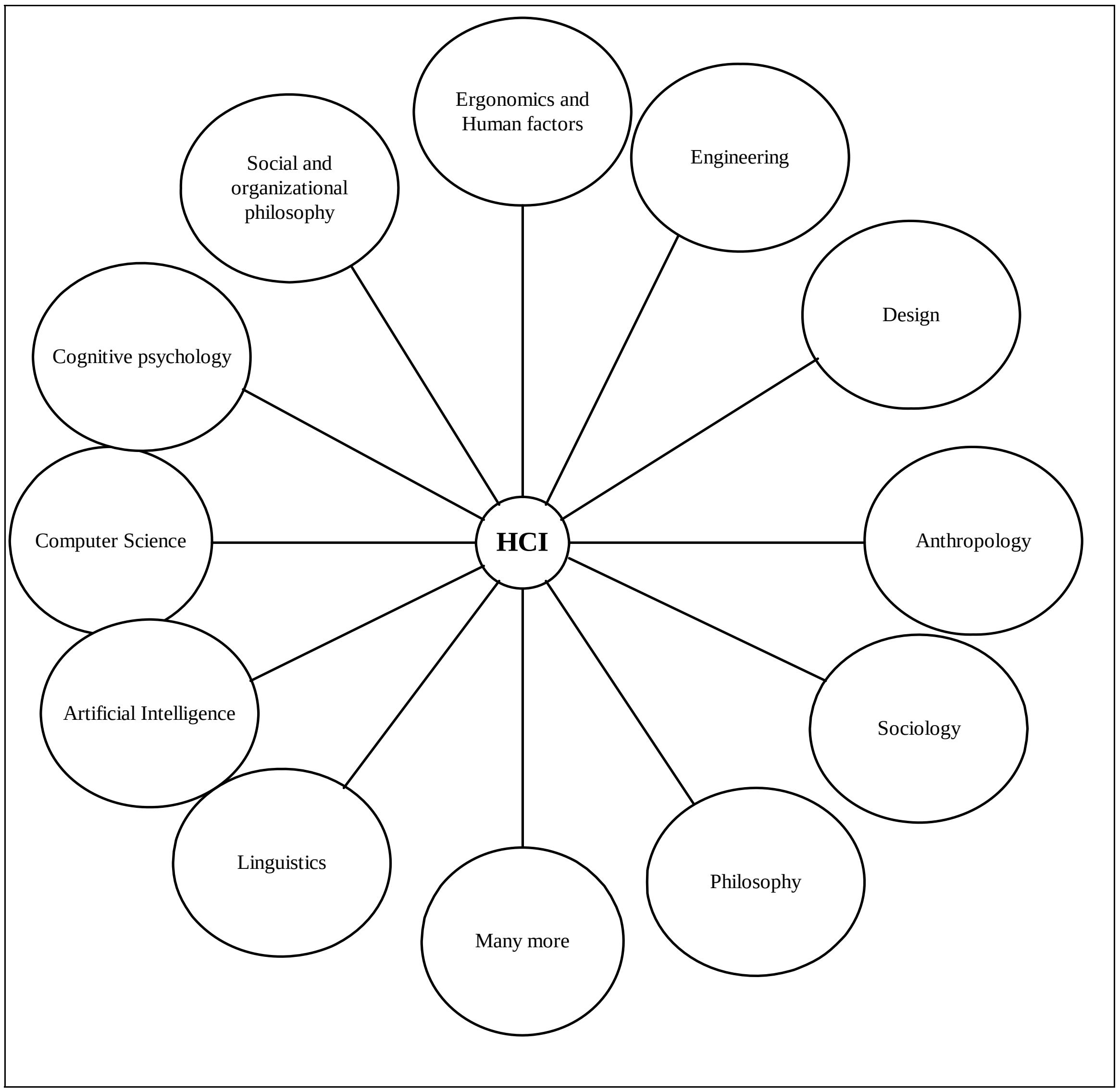


Fig. 1.1 HCI is a multidisciplinary field

The other disciplinary points of view that would place the focus of HCI differently than computer science, just as the focus for a definition of the databases area would be different from a computer science vs. a business perspective. HCI in the large is an interdisciplinary area. It is emerging as a specialty concern within several disciplines, each with different emphases: computer science (application design and engineering of human interfaces), psychology (the application of theories of cognitive processes and the empirical analysis of user behavior), sociology and anthropology (interactions between technology, work, and organization), and industrial design (interactive products). From a computer science perspective, other disciplines serve as supporting disciplines, much as physics serves as a supporting discipline for civil engineering, or as mechanical engineering serves as a supporting discipline for robotics. A lesson learned repeatedly by engineering disciplines is that design problems have a context, and that the overly narrow optimization of one part of a design can be rendered invalid by the broader context of the problem. Even from a direct computer science perspective, therefore, it is advantageous to frame the problem of human-computer interaction broadly enough so as to help students (and practitioners) avoid the classic pitfall of design divorced from the context of the problem.

For example Ergonomics is the Study of the physical characteristics of interaction. This is also known as Human factors. Ergonomics will be good at defining standards and guidelines for constraining the way we design certain aspects of systems. Details about this will be discussed in the proceeding sections. Artificial intelligence is also needed to make the user computer interaction more efficient. The computer system shall be equipped with sufficient artificial intelligence to read the type of human error and supply the necessary feedback. Computer vision is the study and application of methods which allow computers to "understand" image content or content of multidimensional data in general. The term "understand" means here that specific information is being extracted from the image data for a specific purpose: either for presenting it to a human operator or for controlling some process. Study of human psychology is also a very important factor in human computer interaction. By understanding it a programmer may guess the type of user inputs and the possible errors. The term design is a massive term. Regarding HCI, it includes communication design, Graphics design, Information design, Game design etc.

So, the positive part to work in multidisciplinary teams is that more people are involved in doing interaction design, thus more ideas are generated. But the difficult part gets aroused in terms of communication and progress as the designs are created.

1.3.1 Notion of Human

Here Human is actually an end-user which refers to an abstraction of a group of person who will actually use a particular interface. This abstraction is meant to be useful in the process of designing the user interface, and is therefore built on a relevant subset of any user's characteristics which may include what computer interfaces he/she is comfortable with (having used them before or because of their inherent simplicity), his/her technical expertise and degree of knowledge in specific fields or disciplines, and any other information which is believed to be relevant in a specific project. So the human referred here is used in different flavors. These are as follows:

- Human is a classical user i.e. having a general knowledge on usage of computer by gathering experience over previous exposure. For them the user interface can be more detailed having more functionality. Examples of classical users are students, bank manager using banking software.

- Human is a specialized user i.e. having little or no background of computers. Previously they had no computer exposure. For example users using ATM may not have adequate computer exposure or a disabled person using particular software for a particular purpose.

- Human is a group of users i.e. more than one user interacting over a software. For example two users having a conversation over a web-based application like a messenger.

- Human is an organization i.e. computer aided communication among humans, or the nature of the work being cooperatively performed by means of the system. Example: banking software.

1.3.2 Notion of Computer

Computers are generally in the form of desktop PCs or workstations. Instead of workstations, computers may be in the form of embedded computational machines, such as parts of spacecraft cockpits or microwave ovens. Because the techniques for designing these interfaces bear so much relationship to the techniques for designing workstations interfaces, they can be profitably treated together. Computer can also be in the form of Network of Computers. A robot can also be a computer to whom we give commands and expect desired results. Human-computer interaction, by contrast, studies both the mechanism side and the human side, but of a narrower class of devices.

1.3.3 Notion of Interaction

Interaction is a kind of action which occurs as two or more objects have an effect upon one another. The idea of a two-way effect is essential in the concept of interaction instead of a one-way causal effect. An example of interaction may be the feedback during operation of a machines such as a computer or a tool, for example the interaction between a driver and the position of his or her car on the road: by steering the driver influences this position, by looking this information returns to the driver.

A basic goal of HCI is to improve the interaction between users and computers by making computers more user-friendly 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

1.4 HCI – A three-fold discipline

So, this discipline is concerned with three phases of interactive computing systems for human use viz.

- Design
- Implementation
- Evaluation

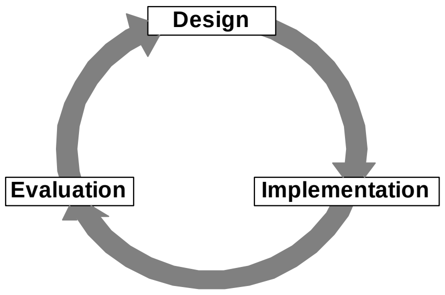


Fig. 1.2 Three major tasks in HCI design

In computer science, interactive computing refers to software which accepts input from humans -- for example, data or commands. Interactive software includes most popular programs, such as word processors or spreadsheet applications. By comparison, non-interactive programs operate without human contact; examples of these include compilers and batch processing applications. If the response is complex enough it is said that the system is conducting social interaction and some systems try to achieve this through the implementation of social interfaces.

2. History of HCI Technology

The history of computer development is often referred to in reference to the different generations of computing devices. Each generation of computer is characterized by a major technological development that fundamentally changed the way computers operate, resulting in increasingly smaller, cheaper, and more powerful and more efficient and reliable devices. Read about each generation and the developments that led to the current devices that we use today. The scope has been shifted from System centered computing (which characterized no interaction and computer comprised of Hardware and machine level code) to people centered computing (which characterized very high level interaction and computer comprises of hardware, software and algorithms).

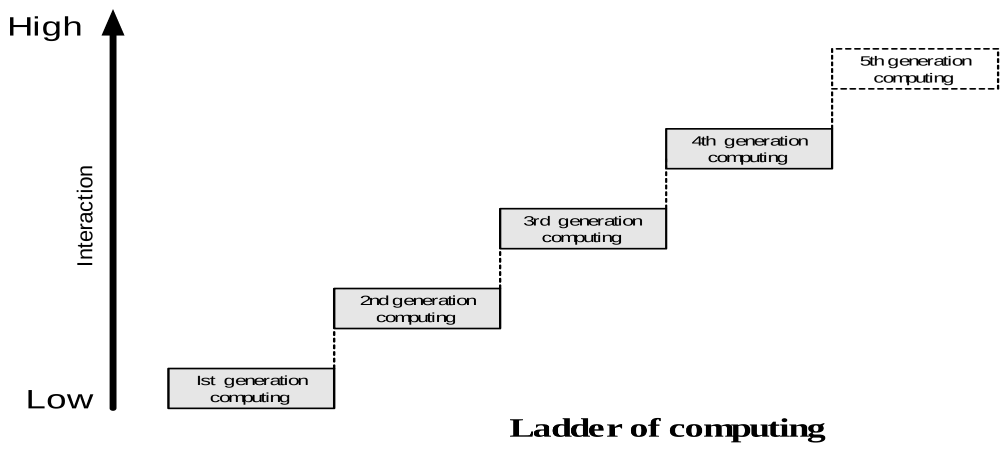


Fig. 1.3 HCI and its evolution

The first generation computers had a very large gap between man and machine. There was machine level programming. At the 2nd generation the communication was at mnemonic level i.e. Assembly language programming (microprocessor era). At the 3rd level generation the gap was reduced to algorithmic level i.e. High-level programming (Software era). At the 3rd generation the gap was very much minimized to intelligence-level i.e. Automatic programming (natural language processing (embedded era). Finally at the 5th level generation the human-computer communication is at human-level i.e. Cognition, perception, psychology, human factors based computation (HCI era).

2.1 0th Generation Computers

Up until the outbreak of the Second World War, computing devices were mechanical or electro-mechanical. The first "known" mechanical calculator (aside from the abacus) was Wilhelm Schickard's "Calculating Clock" (1623). Gottfried Wilhelm von Leibniz also invented a calculator, one that could also multiply (using repeated addition). Charlies Babbage, a 19th century "polymath" devised a calculating machine, the Difference Engine I (1823) which could be used to mechanically generate mathematical tables.

The Mechanical Calculators include "Calculating Clock" (1623) by Wilhelm Schickard, Charles Babbage's Difference Engine (1823) and Analytic Engine.

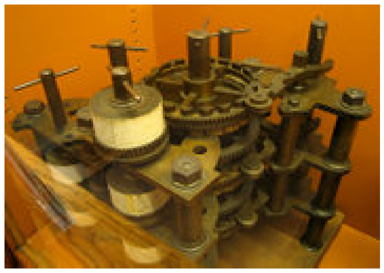
The first multi-purpose, i.e. programmable, computing device was probably Charles Babbage's Difference Engine, which was begun in 1823 but never completed. Lets not go deep into the inventions but rest on the human computer interaction. As from the picture of the difference engine on the right it can be estimated that how complicated it was to interact between the end-user to the computing machine. The mode of interaction was in mechanical level. After this comes the first generation computer where the interaction is machine language level.	 Charles Babbage's Difference Engine

Fig. 1.4 HCI in its first computing concepts

2.2 1st Generation Computers (1940 – 1950)

The first computers used vacuum tubes for circuitry and magnetic drums for memory, and were often enormous, taking up entire rooms. They were very expensive to operate and in addition to using a great deal of electricity, generated a lot of heat, which was often the cause of malfunctions. First generation computers relied on machine language to perform operations, and they could only solve one problem at a time. Input was based on punched cards and paper tape, and output was displayed on printouts.

- **First Generation Electronic Computers (1937-1953) -** Three machines have been promoted at various times as the first electronic computers. These machines used electronic switches, in the form of vacuum tubes, instead of electromechanical relays. In principle the electronic switches would be more reliable, since they would have no moving parts that would wear out, but the technology was still new at that time and the tubes were comparable to relays in reliability. Electronic components had one major benefit, however: they could ``open'' and ``close'' about 1,000 times faster than mechanical switches.

- **Assembly Language Programming -** Software technology during this period was very primitive. The first programs were written out in machine code, i.e. programmers directly wrote down the numbers that corresponded to the instructions they wanted to store in memory. By the 1950s programmers were using a symbolic notation, known as assembly language, then hand-translating the symbolic notation into machine code. Later programs known as assemblers performed the translation task.

2.3 2nd Generation Computers (1950 - 1965)

The second generation saw several important developments at all levels of computer system design, from the technology used to build the basic circuits to the programming languages used to write scientific applications. Electronic switches in this era were based on discrete diode and transistor technology with a switching time of approximately 0.3 microseconds.

Important innovations in computer architecture included index registers for controlling loops and floating point units for calculations based on real numbers. Prior to this accessing successive elements in an array was quite tedious and often involved writing self-modifying code (programs which modified themselves as they ran; at the time viewed as a powerful application of the principle that programs and data were fundamentally the same, this practice is now frowned upon as extremely hard to debug and is impossible in most high level languages). Floating point operations were performed by libraries of software routines in early computers, but were done in hardware in second generation machines.

During this second generation many high level programming languages were introduced, including FORTRAN (1956), ALGOL (1958), and COBOL (1959). Important commercial machines of this era include the IBM 704 and its successors, the 709 and 7094. The latter introduced I/O processors for better throughput between I/O devices and main memory.

The two main features of this generation of computers are as follows:

- Transistors replaced vacuum tubes and ushered in the second generation of computers. The transistor was invented in 1947 but did not see widespread use in computers until the late 50s. The transistor was far superior to the vacuum tube, allowing computers to become smaller, faster, cheaper, more energy-efficient and more reliable than their first-generation predecessors. Though the transistor still generated a great deal of heat that subjected the computer to damage, it was a vast improvement over the vacuum tube. Second-generation computers still relied on punched cards for input and printouts for output.

- Second-generation computers moved from cryptic binary machine language to symbolic, or assembly, languages, which allowed programmers to specify instructions in words. High-level programming languages were also being developed at this time, such as early versions of COBOL and FORTRAN. These were also the first computers that stored their instructions in their memory, which moved from a magnetic drum to magnetic core technology. The first computers of this generation were developed for the atomic energy industry.

2.4 3rd Generation Computers (1965 – 1975)

The third generation brought huge gains in computational power. Innovations in this era include the use of integrated circuits, or ICs (semiconductor devices with several transistors built into one physical component), semiconductor memories starting to be used instead of magnetic cores, microprogramming as a technique for efficiently designing complex processors, the coming of age of pipelining and other forms of parallel processing, and the introduction of operating systems and time-sharing.

In this level, man can communicate with computer in algorithmic level. Early in this third generation Cambridge and the University of London cooperated in the development of CPL (Combined Programming Language, 1963). CPL was, according to its authors, an attempt to capture only the important features of the complicated and sophisticated ALGOL. However, like ALGOL, CPL was large with many features that were hard to learn. In an attempt at further simplification, Martin Richards of Cambridge developed a subset of CPL called BCPL (Basic Computer Programming Language, 1967). In 1970 Ken Thompson of Bell Labs developed yet another simplification of CPL called simply B, in connection with an early implementation of the UNIX operating system.

Instead of punched cards and printouts, users interacted with third generation computers through keyboards and monitors and interfaced with an operating system, which allowed the device to run many different applications at one time with a central program that monitored the memory. Computers for the first time became accessible to a mass audience because they were smaller and cheaper than their predecessors.

2.5 4th Generation Computers (1975 – 1985)

The next generation of computer systems saw the use of large scale integration (LSI - 1000 devices per chip) and very large scale integration (VLSI - 100,000 devices per chip) in the construction of computing elements. At this scale entire processors will fit onto a single chip, and for simple systems the entire computer (processor, main memory, and I/O controllers) can fit on one chip. Gate delays dropped to about 1ns per gate.

Developments in software include very high level languages such as FP (functional programming) and Prolog (programming in logic). These languages tend to use a declarative programming style as opposed to the imperative style of Pascal, C, FORTRAN, et al. In a declarative style, a programmer gives a mathematical specification of what should be computed, leaving many details of how it should be computed to the compiler and/or runtime system. These languages are not yet in wide use, but are very promising as notations for programs that will run on massively parallel computers (systems with over 1,000 processors). Compilers for established languages started to use sophisticated optimization techniques to improve code, and compilers for vector processors were able to vectorize simple loops (turn loops into single instructions that would initiate an operation over an entire vector).

Two important events marked the early part of the third generation: the development of the C programming language and the UNIX operating system, both at Bell Labs. In 1972, Dennis Ritchie, seeking to meet the design goals of CPL and generalize Thompson's B, developed the C language. Thompson and Ritchie then used C to write a version of UNIX for the DEC PDP-11. This C-based UNIX was soon ported to many different computers, relieving users from having to learn a new operating system each time they change computer hardware. UNIX or a derivative of UNIX is now a de facto standard on virtually every computer system.

Here man and machine communicates at intelligence level. In 1981 IBM introduced its first computer for the home user, and in 1984 Apple introduced the Macintosh. Microprocessors also moved out of the realm of desktop computers and into many areas of life as more and more everyday products began to use microprocessors.

As these small computers became more powerful, they could be linked together to form networks, which eventually led to the development of the Internet. Fourth generation computers also saw the development of GUIs, the mouse and handheld devices.

2.6 5th Generation Computers (1985 – 1990)

The development of the next generation of computer systems is characterized mainly by the acceptance of parallel processing. Until this time parallelism was limited to pipelining and vector processing, or at most to a few processors sharing jobs. The fifth generation saw the introduction of machines with hundreds of processors that could all be working on different parts of a single program. The scale of integration in semiconductors continued at an incredible pace - by 1990 it was possible to build chips with a million components - and semiconductor memories became standard on all computers.

Fifth generation computing devices, based on artificial intelligence, are still in development, though there are some applications, such as voice recognition, that are being used today. The use of parallel processing and superconductors is helping to make artificial intelligence a reality. Quantum computation and molecular and nanotechnology will radically change the face of computers in years to come. The goal of fifth-generation computing is to develop devices that respond to natural language input and are capable of learning and self-organization.

The gap between human and computer became very much narrow. Computer communicates with man at human level. Human factors are taken into considerations.

2.7 6th Generation Computers (1990)

This generation is beginning with many gains in parallel computing, both in the hardware area and in improved understanding of how to develop algorithms to exploit diverse, massively parallel architectures. Parallel systems now compete with vector processors in terms of total computing power and most expect parallel systems to dominate the future. One of the most dramatic changes in the sixth generation will be the explosive growth of wide area networking. Network bandwidth has expanded tremendously in the last few years and will continue to improve for the next several years.

In today's high performance computing environment, a computational scientist's routine activities rely heavily on the Internet. Activities include exchange of e-mail and interactive talk or chat sessions with colleagues. Heavy use is made of the ability to transfer documents such as proposals, technical papers, data sets, computer programs and images. A networked high performance computing environment provides the computational scientist access to a wide array of computer architectures and applications. Using telnet to connect to a remote computer (on which one has an account) on the Internet enables the computational scientist to use all of the computational power and software applications of those remote machines.

2.8 A Brief History of Interactions

So far we have discussed the progression of Computer Generations and how the way of interactions have improved with time and technology. Now lets have a snapshot of the important research developments at university, government and corporate research labs. Many of the most famous HCI success developed by companies are deeply rooted in university research. In fact, virtually all of today’s major interface styles and applications have had significant influence from research at universities and labs, often with government funding. Without this research, many of the advances in the field of HCI would probably not have taken place, and as a consequence, the user interfaces of commercial products would be far more difficult to use and learn than they are today. *(Reference. 2)*

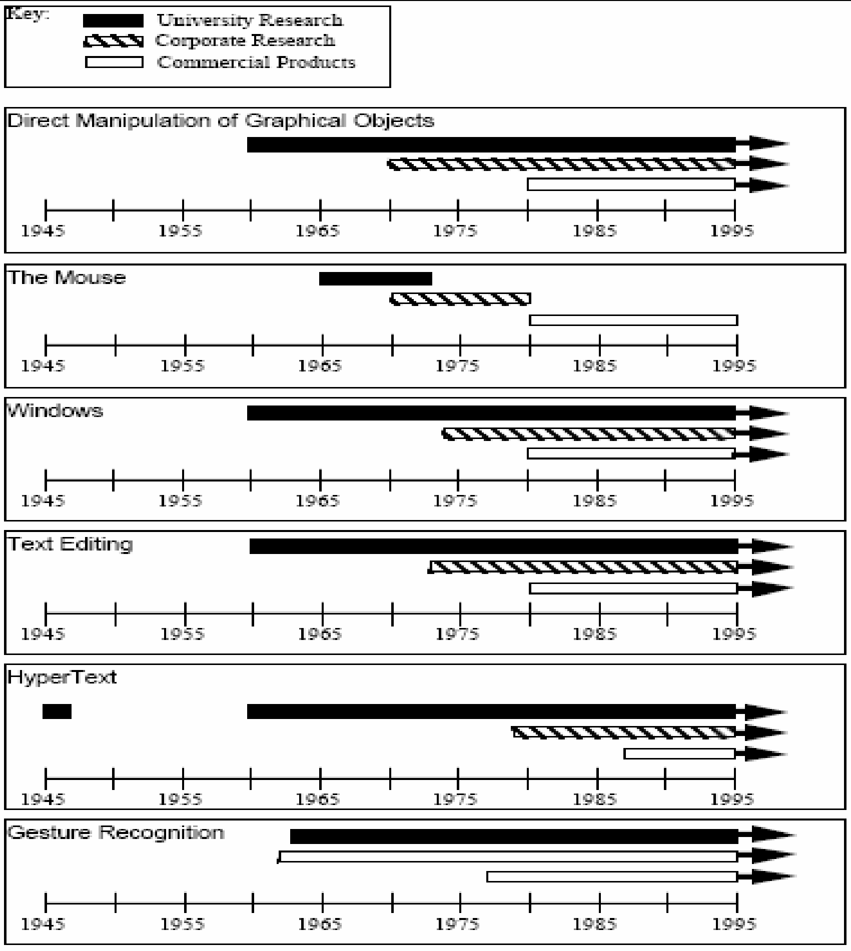


Figure 1: Approximate time lines showing where work was performed on some major technologies discussed in this article.

➢ BASIC INTERACTIONS

Direct Manipulation of Graphical Object

The direct manipulation interface, where visible objects on the screen are directly manipulated with a pointing device, was first demonstrated by Ivan Sutherland in Sketchpad. SketchPad supported the manipulation of objects using a lightpen, including grabbing objects, moving them, changing size, and using constraints. It contained the seeds of myriad important interface ideas. The system was built at Lincoln Labs with support from the Air Force and NSF. William Newman's Reaction Handler, created at Imperial College, London (1966-67) provided direct manipulation of graphics, and introduced "Light Handles," a form of graphical potentiometer, that was probably the first "widget." Another early system was AMBIT/G (implemented at MIT's Lincoln Labs, 1968, ARPA funded). It employed, among other interface techniques, iconic representations, gesture recognition, dynamic menus with items selected using a pointing device, selection of icons by pointing, and moded and modefree styles of interaction. David Canfield Smith coined the term "icons" in his 1975 Stanford PhD thesis on Pygmalion (funded by ARPA and NIMH) and Smith later popularized icons as one of the chief designers of the Xerox Star. Many of the interaction techniques popular in direct manipulation interfaces, such as how objects and text are selected, opened, and manipulated, were researched at Xerox PARC in the 1970's. In particular, the idea of "WYSIWYG" (what you see is what you get) originated there with systems such as the Bravo text editor and the Draw drawing program The concept of direct manipulation interfaces for everyone was envisioned by Alan Kay of Xerox PARC in a 1977 article about the "Dynabook" . The first commercial systems to make extensive use of Direct Manipulation were the Xerox Star (1981), the Apple Lisa (1982) and Macintosh (1984) . Ben Shneiderman at the University of Maryland coined the term "Direct Manipulation" in 1982 and identified the components and gave psychological foundations.

The Mouse

The mouse was developed at Stanford Research Laboratory (now SRI) in 1965 as part of the NLS project (funding from ARPA, NASA, and Rome ADC) to be a cheap replacement for light-pens, which had been used at least since 1954. Many of the current uses of the mouse were demonstrated by Doug Engelbart's as part of NLS in a movie created in 1968. The mouse was then made famous as a practical input device by Xerox PARC in the 1970's. It first appeared commercially as part of the Xerox Star (1981), the Three Rivers Computer Company's PERQ (1981), the Apple Lisa (1982), and Apple Macintosh (1984). *(Reference. 2)*

Windows

Multiple tiled windows were demonstrated in Engelbart's NLS in 1968. Early research at Stanford on systems like COPILOT (1974) and at MIT with the EMACS text editor (1974) also demonstrated tiled windows. Alan Kay proposed the idea of overlapping windows in his 1969 University of Utah PhD thesis and they first appeared in 1974 in his Smalltalk system at Xerox PARC, and soon after in the InterLisp system. Some of the first commercial uses of windows were on Lisp Machines Inc. (LMI) and Symbolics Lisp Machines (1979), which grew out of MIT AI Lab projects. The Cedar Window Manager from Xerox PARC was the first major tiled window manager (1981) followed soon by the Andrew window manager by Carnegie Mellon University's Information Technology Center (1983, funded by IBM). The main commercial systems popularizing windows were the Xerox Star (1981), the Apple Lisa (1982) and most importantly the Apple Macintosh (1984). The early versions of the Star and Microsoft Windows were tiled, but eventually they supported overlapping windows like the Lisa and Macintosh. The X Window System, a current international standard, was developed at MIT in 1984.

➢ APPLICATION TYPES

Drawing Programs

Much of the current technology was demonstrated in Sutherland's 1963 Sketchpad system. The use of a mouse for graphics was demonstrated in NLS (1965). In 1968 Ken Pulfer and Grant Bechthold at the National Research Council of Canada built a mouse out of wood patterned after Engelbart's and used it with a keyframe animation system to draw all the frames of a movie. A subsequent movie, "Hunger" in 1971 won a number of awards, and was drawn using a tablet instead of the mouse (funding by the National Film Board of Canada). William Newman's Markup (1975) was the first drawing program for Xerox PARC's Alto, followed shortly by Patrick Baudelaire's Draw which added handling of lines and curves. The first computer painting program was probably Dick Shoup's "Superpaint" at PARC (1974-75).

Text Editing

In 1962 at the Stanford Research Lab, Engelbart proposed, and later implemented, a word processor with automatic word wrap, search and replace, user definable macros, scrolling text, and commands to move, copy, and delete characters, words, or blocks of text. Stanford's TVEdit (1965) was one of the first CRT-based display editors that was widely used. The Hypertext Editing System from Brown University had screen editing and formatting of arbitrary-sized strings with a lightpen in 1967 (funding from IBM). NLS demonstrated mouse-based editing in 1968. TECO from MIT was an early screen-editor (1967) and EMACS developed from it in 1974. Xerox PARC's Bravo was the first WYSIWYG editorformatter (1974). It was designed by Butler Lampson and Charles Simonyi who had started working on these concepts around 1970 while at Berkeley. The first commercial WYSIWYG editors were the Star, LisaWrite and then MacWrite.

Spreadsheets

The initial spreadsheet was VisiCalc which was developed by Frankston and Bricklin (1977-8) for the Apple II while they were students at MIT and the Harvard Business School. The solver was based on a dependency-directed backtracking algorithm by Sussman and Stallman at the MIT AI Lab.

Hypertext

The idea for hypertext (where documents are linked to related documents) is credited to Vannevar Bush's famous MEMEX idea from 1945. Ted Nelson coined the term "hypertext" in 1965. Engelbart's NLS system at the Stanford Research aboratories in 1965 made extensive use of linking (funding from ARPA, NASA, and Rome ADC). The "NLS Journal" was one of the first on-line journals, and it included full linking of articles (1970). The Hypertext Editing System, jointly designed by Andy van Dam, Ted Nelson, and two students at Brown University (funding from IBM) was distributed extensively. The University of Vermont's PROMIS (1976) was the first Hypertext system released to the user community. It was used to link patient and patient care information at the University of Vermont's medical center. The ZOG project (1977) from CMU was another early hypertext system, and was funded by ONR and DARPA. Ben Shneiderman's Hyperties was the first system where highlighted items in the text could be clicked on to go to other pages (1983, Univ. of Maryland). HyperCard from Apple (1988) significantly helped to bring the idea to a wide audience. There have been many other hypertext systems through the years. Tim Berners-Lee used the hypertext idea to create the World Wide Web in 1990 at the government-funded European Particle Physics Laboratory (CERN). Mosaic, the first popular hypertext browser for the World-Wide Web was developed at the Univ. of Illinois' National Center for Supercomputer Applications (NCSA).*(Reference. 2)*

Computer Aided Design

The same 1963 IFIPS conference at which Sketchpad was presented also contained a number of CAD systems, including Doug Ross's Computer-Aided Design Project at MIT in the Electronic Systems Lab and Coons' work at MIT with SketchPad. Timothy Johnson's pioneering work on the interactive 3D CAD system Sketchpad was his 1963 MIT MS thesis (funded by the Air Force). The first CAD/CAM system in industry was probably General Motor's DAC-1 (about 1963).

Video Games

The first graphical video game was probably SpaceWar by Slug Russel of MIT in 1962 for the PDP-1 including the first computer joysticks. The early computer Adventure game was created by Will Crowther at BBN, and Don Woods developed this into a more sophisticated Adventure game at Stanford in 1966. Conway's game of LIFE was implemented on computers at MIT and Stanford in 1970. The first popular commercial game was Pong (about 1976). *(Reference. 2)*

3. Nature of Human Computer Interaction

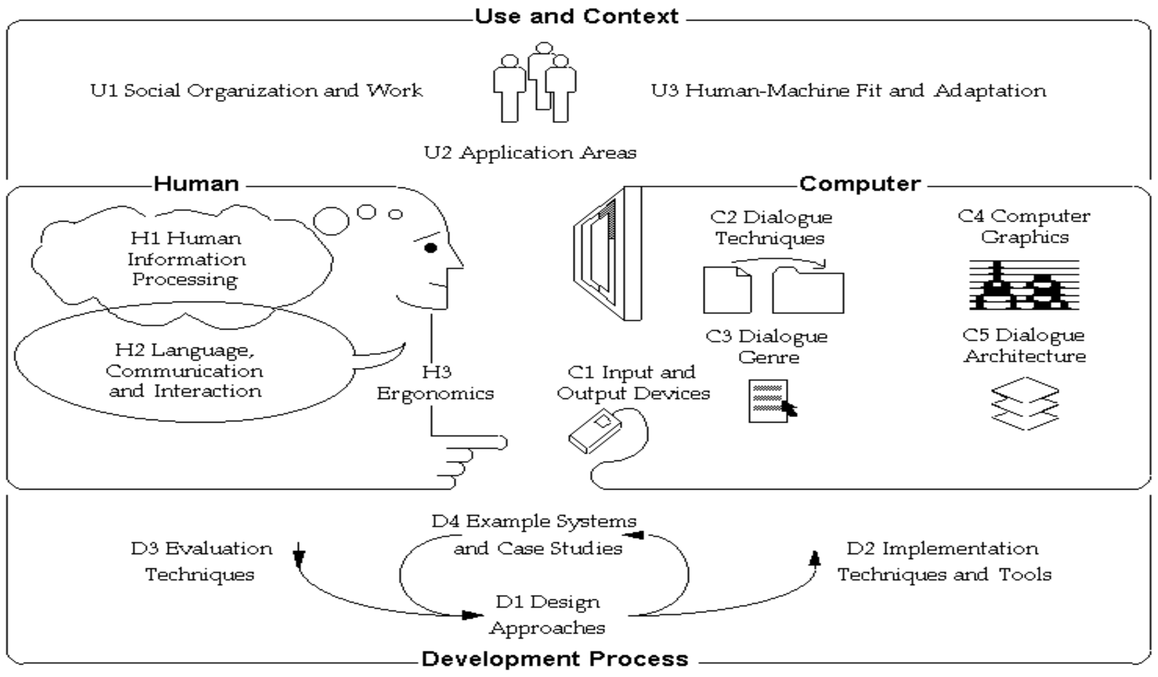

Use and Context
U1 Social Organization and Work
U3 Human-Machine Fit and Adaptation
U2 Application Areas
Human
H1 Human Information Processing
H2 Language, Communication and Interaction
H3 Ergonomics
Computer
C2 Dialogue Techniques
C4 Computer Graphics
C3 Dialogue Genre
C5 Dialogue Architecture
C1 Input and Output Devices
D4 Example Systems and Case Studies
D3 Evaluation Techniques
D2 Implementation Techniques and Tools
D1 Design Approaches
Development Process

Fig. 1.5 Human Computer Interaction *(Reference. 1)*

From the picture we see that the Human Computer Interaction has got mainly 4 sections viz. Use and Context, Human, Computer and Development Process. So let's have a brief description about the respective sections.

3.1 Use and Context of Computers

The uses to which computers are put are spoken of as 'applications' in the computer world. These uses and the extent to which the interface (and the application logic in the rest of the system) fits them can have a profound impact on every part of the interface and its success. Moreover, the general social, work, and business context may be important. In addition to technical requirements, an interface may have to satisfy quality-of-work-life goals of a labor union or meet legal constraints on "look and feel" or position the image of a company in a certain market. The following topics are concerned with general problems of fitting computers, uses, and context of use together. *(Reference. 1)*

3.1.1 Social Organization and Work

This heading relates to the human as an interacting social being. It includes a concern with the nature of work, and with the notion that human systems and technical systems mutually adapt to each other and must be considered as a whole.

- Points of view (e.g., industrial engineering, operations research, Rasmussen's cognitive engineering, the Aarhus participatory design approach, Hewitt's open systems)
- Models of human activity (e.g., opportunistic planning, open procedures)
- Models of small-groups, organizations
- Models of work, workflow, cooperative activity, office work
- Socio-technical systems, human organizations as adaptive open systems, mutual impact of computer systems on work and vice versa, computer systems for group tasks, case studies
- Quality of work life and job satisfaction

3.1.2 Application Areas

The focus of this section is on classes of application domains and particular application areas where characteristic interfaces have developed.

- Characterization of application areas (e.g., individual vs. group, paced vs. unpaced)
- Document-oriented interfaces: Text-editing, document formatting, illustrators, spreadsheets, hypertext
- Communications-oriented interfaces: Electronic mail, computer conferencing, telephone and voice messaging systems
- Design environments: programming environments, CAD/CAM
- On-line tutorial systems and help systems
- Multimedia information kiosks
- Continuous control systems: process control systems, virtual reality systems, simulators, cockpits, video games
- Embedded systems: Copier controls, elevator controls, consumer electronics and home appliance controllers (e.g., TVs, VCRs, microwave ovens, etc.)

3.1.3 Human Machine Fit and Adaptation

Part of the purpose of design is to arrange a fit between the designed object and its use. There are several dimensions to this fit and it is possible to place the burden of adjustment in different places: Adjustments can be made either at design time or at time of use by either changing the system or the user and the changes can be made by either the users themselves or, sometimes, by the system. Topics under this heading all relate to changing some component of a socio-technical system so as to improve its fit.

- Alternate techniques for achieving fit
- Nature of adaptive systems, adaptations of human systems that cancel reliability improvements, the nature of error in adaptive redundant systems, empirical findings on user improvisation with routine systems, determinants of successful systems introduction,
- System selection: theories of system adoption
- System adaptation: customization and tailorability techniques
- User selection: compatibilities of user and system characteristics
- User adaptation: ease of learning, training methods (e.g., on-line tutorials), relation to system design
- User guidance: help techniques, documentation, error-handling techniques

3.2 Human Characteristics

It is important to understand something about human information-processing characteristics, how human action is structured, the nature of human communication, and human physical and physiological requirements.

3.2.1 Characteristics of Human Information Processing

- Models of cognitive architecture: symbol-system models, connectionist models, engineering models
- Phenomena and theories of memory
- Phenomena and theories of perception
- Phenomena and theories of motor skills
- Phenomena and theories of attention and vigilance
- Phenomena and theories of problem solving
- Phenomena and theories of learning and skill acquisition
- Phenomena and theories of motivation
- Users' conceptual models
- Models of human action
- Human diversity, including disabled populations

3.2.2 Language, Communication and Interaction

- Aspects of language: syntax, semantics, pragmatics
- Formal models of language
- Pragmatic phenomena of conversational interaction (e.g., turn-taking, repair)
- Language phenomena
- Specialized languages (e.g., graphical interaction, query, command, production systems, editors)
- Interaction reuse (e.g., history lists)

3.2.3 Ergonomics

Anthropometric and physiological characteristics of people and their relationship to workspace and environmental parameters.

- Human anthropometry in relation to workspace design
- Arrangement of displays and controls, link analysis
- Human cognitive and sensory limits
- Sensory and perceptual effects of CRT and other display technologies, legibility, display design
- Control design
- Fatigue and health issues
- Furniture and lighting design
- Temperature and environmental noise issues
- Design for stressful or hazardous environments
- Design for the disabled

3.3 Computer System and Interface Architecture

Machines have specialized components for interacting with humans. Some of these components are basically transducers for moving information physically between human and machine. Other components have to do with the control structure and representation of aspects of the interaction. These specialized components are covered in the following topics.

3.3.1 Input and Output Devices

The technical construction of devices for mediating between humans and machines.

- Input devices: survey, mechanics of particular devices, performance characteristics (human and system), devices for the disabled, handwriting and gestures, speech input, eye tracking, exotic devices (e.g., EEG and other biological signals)
- Output devices: survey, mechanics of particular devices, vector devices, raster devices, frame buffers and image stores, canvases, event handling, performance characteristics, devices for the disabled, sound and speech output, 3D displays, motion (e.g., flight simulators), exotic devices
- Characteristics of input/output devices (e.g., weight, portability, bandwidth, sensory modality)
- Virtual devic

3.3.2 Dialogue Techniques

The basic software architecture and techniques for interacting with humans.

Dialogue Inputs:

- Types of input purposes (e.g., selection, discrete parameter specification, continuous control)
- Input techniques: keyboard techniques (e.g, commands, menus), mouse-based techniques (e.g., picking, rubber-band lines), pen-based techniques (e.g., character recognition, gesture), voice-based techniques

Dialogue Outputs:

- Types of output purposes (e.g., convey precise information, summary information, illustrate processes, create visualizations of information)
- Output techniques (e.g., scrolling display, windows, animation, sprites, fish-eye displays)
- Screen layout issues (e.g., focus, clutter, visual logic)

Dialogue Interaction Techniques:

- Dialogue type and techniques (e.g., alphanumeric techniques, form filling, menu selection, icons and direct manipulation, generic functions, natural language)
- Navigation and orientation in dialogues, error management
- Multimedia and non-graphical dialogues: speech input, speech output, voice mail, video mail, active documents, videodisc, CD-ROM
- Agents and AI techniques
- Multi-person dialogues

Dialogue Issues:

- Real-time response issues
- Manual control theory
- Supervisory control, automatic systems, embedded systems
- Standards
- "Look and feel," intellectual property protection

3.3.3 Dialogue Genre

The conceptual uses to which the technical means are put. Such concepts arise in any media discipline (e.g., film, graphic design, etc.).

- Interaction metaphors (e.g., tool metaphor, agent metaphor)
- Content metaphors (e.g., desktop metaphor, paper document metaphor)
- Persona, personality, point of view
- Workspace models
- Transition management (e.g., fades, pans)
- Relevant techniques from other media (e.g., film, theater, graphic design)
- Style and aesthetics

3.3.4 Computer Graphics

Basic concepts from computer graphics that are especially useful to know for HCI.

- Geometry in 2- and 3- space, linear transformations
- Graphics primitives and attributes: bitmap and voxel representations, raster-op, 2-D primitives, text primitives, polygon representation, 3-D primitives, quadtrees and octtrees, device independent images, page definition languages
- Solid modeling, splines, surface modeling, hidden surface removal, animation, rendering algorithms, lighting models
- Color representation, color maps, color ranges of devices

3.3.5 Dialogue Architecture

Software architectures and standards for user interfaces.

- Layers model of the architecture of dialogues and windowing systems, dialogue system reference models
- Screen imaging models (e.g., RasterOp, Postscript, Quickdraw)
- Window manager models (e.g., Shared address-space, client-server), analysis of major window systems (e.g., X, New Wave, Windows, Open Look, Presentation Manager, Macintosh)
- Models of application-to-dialogue manager connection
- Models for specifying dialogues
- Multi-user interface architectures "Look and feel"
- Standardization and interoperability

3.4 Development Process

The construction of human interfaces is both a matter of design and engineering. These topics are concerned with the methodology and practice of interface design. Other aspects of the development process include the relationship of interface development to the engineering (both software and hardware) of the rest of the system.

3.4.1 Design Approaches

The process of design. Relevant topics from other design disciplines.

- Graphic design basics (e.g., design languages, typography, use of color, 2D & 3D spatial organization, temporal sequencing, etc.)
- Alternative system development processes (e.g., waterfall model, participatory design), lifecycle model, iterative design, choice of method under time/resource constraint
- Task analysis techniques (e.g., field studies, analytical methods), task allocation, market analysis

- Design specification techniques
- Design analysis techniques (e.g., objects and actions)
- Industrial design basics, case studies and empirical analysis of design

3.4.2 Implementation Techniques and Tools

Tactics and tools for implementation.

- Relationships among design, evaluation, and implementation
- Independence and reusability, application independence, device independence
- Prototyping techniques (e.g., storyboarding, video, "Wizard of Oz", HyperCard, rapid prototype implementations)
- Dialogue toolkits (e.g., MacApp, NextStep, UIMS's, HyperCard)
- Object-oriented methods
- Data representation and algorithms

3.4.3 Evaluation Techniques

Philosophy and specific methods for evaluations.

- Productivity
- Figures of merit (e.g., time, errors, learnability, design for guessing, preference, etc.)
- Usability testing techniques, linking testing to specifications
- Formative and summative evaluation techniques for empirical evaluation, including, field observation methods, participant observation, interviewing techniques, questionnaire design, psychometric methods, video protocols, system logging, experiment design (e.g, concern with sample bias, etc.), methods from psychological and sociological evaluation fields, ethics of working with participants

3.4.4 Example Systems and Case Studies

Classic designs to serve as extended examples of human interface design.

Command-oriented:

- OS/360 JCL (batch-oriented command style, baseline for seeing later improvements)
- PC DOS (command style interface learned by millions)
- Airline check-in system (time pressure, ambiguous input, distributed system)

Graphics-oriented:

- Xerox Star (icon-window interface, generic commands)
- Apple Macintosh (similar interface over many applications)
- MacPaint (widely known and available graphics program)

Frame-based:

- Promis (Rapid response to large set of frames, touch-panel oriented)
- Zog (User-tailorable, rapid-response system, large number of frames, first commercial frame-based system)
- HyperCard (Graphically-oriented frame-based system with user programming language, first mass market frame-oriented system).

User-defined combinatorics:

- Unix operating system (strong combinatoric architecture paired with weak human factors)
- Emacs (language-oriented, large combinatoric command set)
- Visicalc (a "home-run" application with strong conceptual model that succeeded despite weak human factors)
- DBaseIII (simple, but successful, user applications generator)
- Interfaces for untrained, walk-up users:
- Olympic Message System (practical use of user testing under time pressure)
- Nintendo Super Mario Brothers (learnable without a manual by grade school children)

4. Future Human Computer Interaction

Machines have specialized components for interacting with humans. Some of these components are basically transducers for moving information physically between human and machine. Other components have to do with the control structure and representation of aspects of the interaction. As we have seen earlier how the nature of interactions have enhanced. Now let's have a look on the future aspects of HCI.

4.1 Moore's Law

"...The first microprocessor only had 22 hundred transistors. We are looking at something a million times that complex in the next generations—a billion transistors. What that gives us in the way of flexibility to design products is phenomenal."

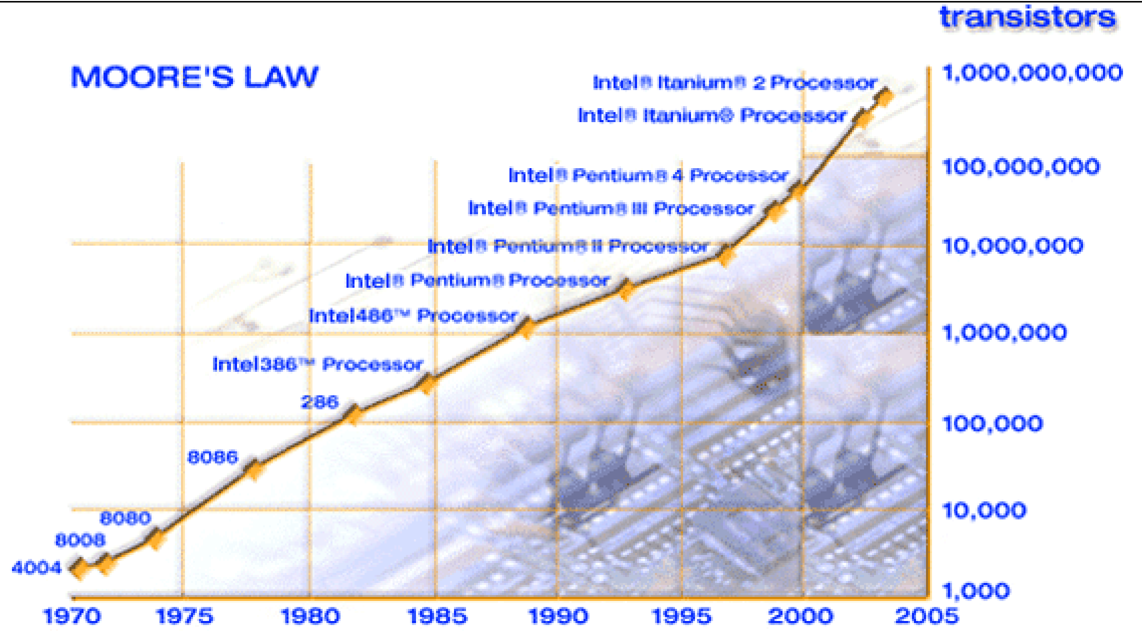
—Gordon E. Moore

In 1965, Intel co-founder Gordon Moore saw the future. His prediction, popularly known as Moore's Law, states that the number of transistors on a chip doubles about every two years. This observation about silicon integration, made a reality by Intel, has fueled the worldwide technology revolution. Moore predicted that this trend would continue for the foreseeable future. In subsequent years, the pace slowed down a bit, but data density has doubled approximately every 18 months, and this is the current definition of Moore's Law, which Moore himself has blessed. Most experts, including Moore himself, expect Moore's Law to hold for at least another two decades

Our world-leading silicon technologies have supported the development and worldwide adoption of our industry-standard architectures and platforms, making Intel the world's largest silicon supplier. The best is yet to come. Our R&D investment and silicon expertise support unique Intel breakthroughs that will enable us to drive Moore's Law well into the future and deliver more exciting capabilities into our technologies. Intel's unparalleled silicon expertise gives us an edge on the competition in developing leading architectures and platforms that will continue fueling economic growth and improving the lives of billions of people throughout the world.

More Performance for Less Cost

Many are familiar with Intel's exponential increases in the number of transistors integrated into our processors and other leading platform ingredients. These increases, as the following graph illustrates, have steadily and reliably lead to more computing performance as measured in millions of instructions per second (MIPS).



Moore's Law Means More Performance. Processing power, measured in millions of instructions per second (MIPS), has risen because of increased transistor counts.

Fig. 1.6 Moore's law: Arguing the feasibility of HCI

But Moore's Law also means decreasing costs. As silicon-based components and platform ingredients gain in performance, they become exponentially cheaper to produce, and therefore more plentiful, more powerful, and more seamlessly integrated into our daily lives. Today's microprocessors run everything from toys to traffic lights. A musical birthday card costing a few U.S. dollars today has more computing power than the fastest mainframes of a few decades ago.

4.2 Human abilities and Compuman

With time human intelligence is also increasing. People are more computer knowledged than previous ones. So the human computer interaction can be more sophisticated. We can imagine the computer to be a compuman matching its attributes to that of a man.

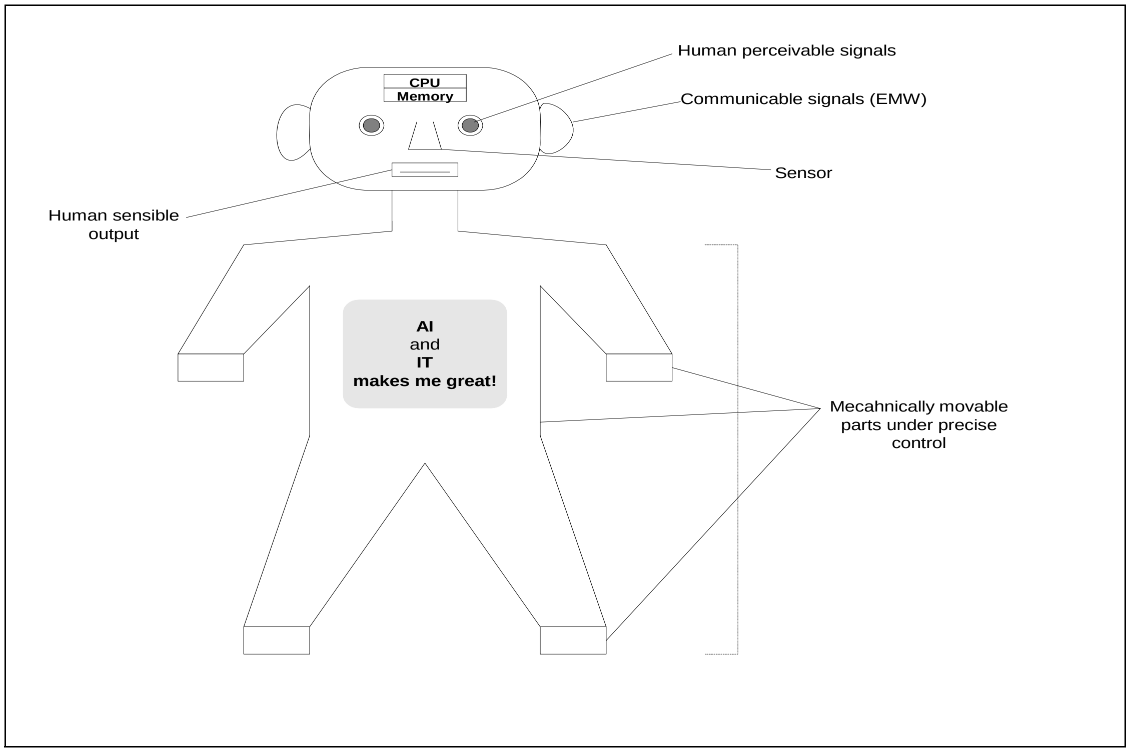

Human perceivable signals
CPU
Memory
Communicable signals (EMW)
Sensor
Human sensible output
AI and IT makes me great!
Mecahnically movable parts under precise control

Fig. 1.7 Compuman: Towards the reality of future HCI

The CPU and Memory of a computer can be compared to that of Human brain. There should be communicable signals as well as human sensible output and sensor. These entire if incorporated in a computer then the human computer interaction will be more efficient. The processing speed of the compuman should be that of a computer, memory capacity should be that of a computer but it should also have human factors like mind, mood, emotion, perception, cognition, psychology etc. Thus with all these attributes a computer can behave very close to Human. In future behavior of a human can be more accurately and efficiently incorporated or simulated in a computer.

4.3 Up-coming Areas

Gesture Recognition: The first pen-based input device, the RAND tablet, was funded by ARPA. Sketchpad used light-pen gestures (1963). Teitelman in 1964 developed the first trainable gesture recognizer. A very early demonstration of gesture recognition was Tom Ellis' GRAIL system on the RAND tablet (1964, ARPA funded). It was quite common in light-pen-based systems to include some gesture recognition, for example in the AMBIT/G system (1968 — ARPA funded). A gesture-based text editor using proofreading symbols was developed at CMU by Michael Coleman in 1969. Bill Buxton at the University of Toronto has been studying gesture-based interactions since 1980. Gesture recognition has been used in commercial CAD systems since the 1970s, and came to universal notice with the Apple Newton in 1992.

Multi-Media: The FRESS project at Brown used multiple windows and integrated text and graphics (1968, funding from industry). The Interactive Graphical Documents project at Brown was the first hypermedia (as opposed to hypertext) system, and used raster graphics and text, but not video (1979-1983, funded by ONR and NSF). The Diamond project at BBN (starting in 1982, DARPA funded) explored combining multimedia information (text, spreadsheets, graphics, speech). The Movie Manual at the Architecture Machine Group (MIT) was one of the first to demonstrate mixed video and computer graphics in 1983 (DARPA funded).

3-D: The first 3-D system was probably Timothy Johnson's 3-D CAD system mentioned above (1963, funded by the Air Force). The "Lincoln Wand" by Larry Roberts was an ultrasonic 3D location sensing system, developed at Lincoln Labs (1966, ARPA funded). That system also had the first interactive 3-D hidden line elimination. An early use was for molecular modeling. The late 60's and early 70's saw the flowering of 3D raster graphics research at the University of Utah with Dave Evans, Ivan Sutherland, Romney, Gouraud, Phong, and Watkins, much of it government funded. Also, the military-industrial flight simulation work of the 60's - 70's led the way to making 3-D real-time with commercial systems from GE, Evans&Sutherland, Singer/Link (funded by NASA, Navy, etc.). Another important center of current research in 3-D is Fred Brooks' lab at UNC . *(Reference. 2)*

Virtual Reality and "Augmented Reality": The original work on VR was performed by Ivan Sutherland when he was at Harvard (1965-1968, funding by Air Force, CIA, and Bell Labs). Very important early work was by Tom Furness when he was at Wright-Patterson AFB. Myron Krueger's early work at the University of Connecticut was influential. Fred Brooks' and Henry Fuch's groups at UNC did a lot of early research, including the study of force feedback (1971, funding from US Atomic

Energy Commission and NSF). Much of the early research on head-mounted displays and on the DataGlove was supported by NASA.

Natural language and speech: The fundamental research for speech and natural language understanding and generation has been performed at CMU, MIT, SRI, BBN, IBM, AT&T Bell Labs and BellCore, much of it government funded. See, for example, for a survey of the early work.

Quantum Computing: In a quantum computer, the fundamental unit of information (called a quantum bit or *qubit*), is not binary but rather more quaternary in nature. This qubit property arises as a direct consequence of its adherence to the laws of quantum mechanics which differ radically from the laws of classical physics. A qubit can exist not only in a state corresponding to the logical state 0 or 1 as in a classical bit, but also in states corresponding to a blend or *superposition* of these classical states. In other words, a qubit can exist as a zero, a one, or simultaneously as both 0 and 1, with a numerical coefficient representing the probability for each state. This may seem counterintuitive because everyday phenomenons are governed by classical physics, not quantum mechanics -- which takes over at the atomic level. This rather difficult concept is perhaps best explained through an experiment.

Molecular Electronics: The guiding principle of this research is that biological systems can provide useful paradigms for developing electronic and computational devices at the molecular level. For example, natural photosynthetic reaction centers are photovoltaic devices of molecular dimensions, and the principles dictating the operation of reaction centers may be useful in the design of synthetic optoelectronic switches. In this project, several classes of molecular photovoltaic species are being synthesized and studied. These include porphyrin-fullerene dyads, carotenoid-fullerene dyads, a carotenoid-porphyrin-fullerene triad, carotene-porphyrin-imide triads, and molecular dyads and triads containing two porphyrin moieties. The approach involves the design and synthesis of dyads, triads and other supermolecular species using the techniques of organic chemistry. The newly-prepared molecules are then studied by a variety of physical methods, including timeresolved laser spectroscopy, nmr spectroscopy, and cyclic voltammetry in order to determine how, and how well they functioned as molecular electronic elements. The information gained can then be used to design new generations of these molecules. Once functional molecular photovoltaics, logic gates, or other elements have been prepared, ways must be developed for interfacing these with electronic circuits. Possibilities are being investigated in a collaborative project with Professor Michael Kozicki, in the Department of Electrical Engineering.

DNA Computer: Computer chip manufacturers are furiously racing to make the next microprocessor that will topple speed records. Sooner or later, though, this competition is bound to hit a wall. Microprocessors made of silicon will eventually reach their limits of speed and miniaturization. Chip makers need a new material to produce faster computing speeds. Scientists have found the new material they need to build the next generation of microprocessors. Millions of natural supercomputers exist inside living organisms, including your body. DNA (deoxyribonucleic acid) molecules, the material our genes are made of, have the potential to perform calculations many times faster than the world's most powerful human-built computers. DNA might one day be integrated into a computer chip to create a so-called biochip that will push computers even faster. DNA molecules have already been harnessed to perform complex mathematical problems. While still in their infancy, DNA computers will be capable of storing billions of times more data than your personal computer. In this article, you'll learn how scientists are using genetic material to create nano-computers that might take the place of silicon-based computers in the next decade.

Computer Supported Cooperative Work. Doug Engelbart's 1968 demonstration of NLS included the remote participation of multiple people at various sites (funding from ARPA, NASA, and Rome ADC). Licklider and Taylor predicted on-line interactive communities in an 1968 article and speculated about the problem of access being limited to the privileged. Electronic mail, still the most widespread multiuser software, was enabled by the ARPAnet, which became operational in 1969 and by the Ethernet from Xerox PARC in 1973. An early computer conferencing system was Turoff's EIES system at the New Jersey Institute of Technology (1975).

5. Discussion

Previously the acronym CHI, for Computer-Human Interaction, has been used to refer this field. However, researchers and practitioners now refer to their field of study as HCI, which perhaps rose in popularity partly because of the notion that the human, and the human's needs and time, should be considered first, and are more important than the machine's. This notion became increasingly relevant towards the end of the 20th century as computers became increasingly inexpensive (as did CPU time), small, and powerful. Since the turn of the millennium, the field of human-centered computing has emerged as an even more pronounced focus on understanding human beings as actors within socio-technical systems. Design methodologies in HCI aim to create user interfaces that are usable, i.e. that can be operated with ease and efficiency. However, an even more basic requirement is that the user interface be useful, i.e. that it allow the user to complete relevant tasks. Software products are often touted by marketers as being "intuitive" and "natural" to use, often simply because they have a graphical user interface. Many researchers in HCI view such claims as unfounded (e.g. a poorly designed GUI may be very unusable), and some object to the use of the words intuitive and natural as vague and/or misleading, since these are very context-dependent terms.

Now designs are made user centered. User-centered design is now 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.

6. References

1: http://sigchi.org/cdg/index.html

2: "A Brief History of Human Computer Interaction Technology"
- Brad A. Myers, Carnegie Mellon University

3: Power Point Presentation by Alan Dix, Lancaster University

4: Power Point Presentation of PrececIDbook by Yvonne Rogers

5: "Human Computer Interaction" by Dr. Keith Andrews, Graz University

4: Power Point Presentation by Dr. Debasis Samanta, IIT Kharagpur