Initially, programs were written exclusively in machine language until compilers and higher level programming languages, such as COBOL (Common Business Oriented Language) and FORTRAN, were developed. The COBOL language was designed to use English statements and be self-documenting. It was specifically intended for business and accounting, and was first introduced in 1960. Grace Hopper created FLOWMATIC, its immediate predecessor, and is considered the "mother" of COBOL (LegacyJ Corp., 2001). FORTRAN, on the other hand was developed specifically for the programmer/scientist at IBM and was first announced in 1957 for the IBM 704. FORTRAN made it possible to code very sophisticated mathematical models and calculations without resorting to assembly language, but was somewhat slow to be widely adopted (Wilkes, 1993).

From the point of view of HCI, the most significant developments of this period were the first visions of what the world of computing might be like, if only we were not limited to punched cards and batch processing on monolithic machines. The five visionaries that stand out during this period were Vannevar Bush, J. C. R. Licklider, Douglas Englebart, Ivan Sutherland, and Ted Nelson.

Vannevar Bush

Vannevar Bush was the Director of the Office of Scientific Research and Development and served as an advisor to Presidents Truman and Roosevelt. At the end of World War II, he wrote an article for the *Atlantic Monthly* in which he summarized his thinking and sought to redirect the attention of scientists from the instrumentalities of war to the need for improved storage, management, and access to information, both scientific and otherwise (Bush, 1945).

> The summation of human experience is being expanded at a prodigious rate, and the means we use for threading through the consequent maze to the momentarily important item is the same as was used in the days of square-rigged ships. (p. 102)

He carefully documented his vision for the kinds of technology that he thought could contribute to the solution of this problem, including photocell-based electronic cameras mounted on one's forehead with which to document what was of particular interest, dry photography, speech recording that could be time-stamped to synchronize it with the pictures being recorded by his advanced camera, electronic calculating machines that could arithmetically *and logically* manipulate data, and speech-controlled typewriters. However, his most interesting ideas were conceptual, relating to the processes of information retrieval:

> Selection by association, rather than indexing, may yet be mechanized. One cannot hope thus to equal the speed and flexibility with which the mind follows an associative trail, but it should be possible to beat the mind decisively in regard to the permanence and clarity of the items resurrected from storage. (p. 104)

Thus, Bush was advocating that we should organize and manipulate information in the same ways we think about it. He used the term Memex to describe a mechanized, associatively indexed, specialized workstation from which to access private files and libraries, complete with screens that projected automatically retrieved microfilm images. Ted Nelson would later call this nonsequential, associative method of cataloging "hypertext" (Nelson, 1965).

J. C. R. Licklider

In 1959, J. C. R. Licklider purchased Bolt Beranek and Newman.[2] (BBN)—its first computer. It was a Royal McBee LGP 30. Having been influenced by the MIT Whirlwind computer and the Lincoln Lab TX-0 and TX-2, it was not an IBM 650, or even a 704, the dominant machines in use at the time, particularly in accounting departments or scientific laboratories. Lick, as he was called by anyone who knew him, was not interested in batch processing and punched cards. He was interested in how individuals could use computers to support their work. The LGP-30 received its input by reading a punched paper tape that was prepared offline by a Flexowriter, a special typewriter that looked a little like a teletype machine. Although the LGP-30 cost $25,000, it was a small machine by today's standards. It had a magnetic drum memory of 8,000 characters and was coded in a primitive machine language. One full rotation of the drum took approximately 1 ms, so that if processing speed or efficiency was of interest, the programmer had to plan the sequence of literal memory locations assigned in the program so that sequential read instructions would not consume extra drum rotations. The machine was definitely one a single user could get his or her arms around, but it was marginally useful and prone to errors, and Lick soon replaced it with Digital Equipment Corporation PDP-1, serial number 1. The PDP-1 was about the capacity of a Radioshack TRS 80, of 1970s vintage, but was attractive because it featured a Cathode-Ray Tube display, and a keyboard, light pen, and paper tape reader as input devices. Even though it cost more than $120,000, it was truly a computer with which an individual could interact.

The year 1960 was also the year Lick published his seminal paper, "Man–Computer Symbiosis" (Licklider, 1960). In it, he laid out his vision for what HCI could be.

> Man-computer symbiosis . . . will involve very close coupling between the human and the electronic member of the partnership. The main aims are 1) to let computers facilitate formulative thinking as they now facilitate the solution of formulated problems, and 2) to enable men and

[2]BBN, originally Bolt Beranek and Newman, is a commercial company that was founded by and named for Richard Bolt, Leo Beranek, and Robert Newman, three MIT academicians whose specialty was acoustics. In 1994, the name was officially shortened to BBN; in 1997, it was acquired by GTE and renamed BBN Technologies. As a result of GTE's merger in 2000 with Bell Atlantic, it is now a business unit of Verizon Communications. Licklider was hired by Leo Beranek because of his interest in psychoacoustics, but, as will become evident, his interests were much broader and he launched BBN into the computer age.

computers to cooperate in making decisions and controlling complex situations without inflexible dependence on predetermined programs. In the anticipated symbiotic partnership, men will set the goals, formulate the hypotheses, determine the criteria and perform the evaluations. Computing machines will do the routinizable work that must be done to prepare the way for insights and decisions in technical and scientific thinking. (Licklider, 1960, Summary, p. 4)

Lick wanted to think of computers as aids to human thinking, not as monolithic, number-crunching machines. He laid out the technological advances required to achieve these goals—developments in:

1. Computer time-sharing, because use of one machine for one knowledge worker was not, at the time, cost-effective.
2. Hardware memory requirements, because he foresaw the need for the user to have access to large quantities of data and reference material, a virtual library at one's fingertips.
3. Memory organization, because serial search through sequentially organized databases was too time-consuming and inefficient.
4. Programming languages, because of the extreme mismatch between languages the computer could understand and those the human could understand.
5. Input–output equipment, because he envisioned the time when input and output should match the "flexibility and convenience of the pencil and doodle pad or the chalk and blackboard" (Licklider, 1960).

In this vein, he also discussed the then-current state of speech recognition technology (claimed to be 98% correct recognition of decimal digits at Bell Laboratories and MIT Lincoln Labs.) and the need for computer-posted wall displays. He was well aware of the fledgling state of artificial intelligence (AI) research. He cited an Air Force study group conclusion that it would be at least 1980, that is 20 years hence, before developments in AI would make it possible for machines alone to do much thinking or problem-solving of military significance. To that, Lick added,

That would leave, say, five years to develop man-computer symbiosis and 15 years to use it. The 15 may be 10 or 500, but those years should be intellectually the most creative and exciting in the history of mankind.

In 1962, Lick became the Director of the Information Processing Techniques Office at the Advanced Research Projects Agency (ARPA) of the Department of Defense and actually served during two periods of time; 1962–1964 and 1974–1975. In that role, he developed a far-reaching program of contract research in all of the elements required to achieve his vision. By 1963, that vision had broadened to include the concept of computer networking and the Internet. In a memo in 1963 to his ARPA research community, addressed as "the members and affiliates of the Intergelactic Computer Network," he outlined an important piece of his strategy—to connect all the community's computers and time-sharing systems into a single computer network spanning the continent. Then, in 1965, in a book entitled *Libraries of the Future* (Licklider, 1965), he expanded on this idea:

The concept of a "desk" may have changed from passive to active: a desk may be primarily a display-and-control station in a telecommunication-telecomputation system and its most vital part may be the cable ("umbilical cord") that connects it, via a wall socket, into the precognitive utility net. (Licklider, 1965, p. 33)

Lick passed away in 1990 at age 75. Today, he is widely credited with having been the first to have a vision of computer networking and the Internet. For a comprehensive record of Lick's Contributions see Waldrop (2001).

Douglas Englebart

One of the key contributors to the ARPA program that Lick originated was Douglas Englebart. In 1950, Englebart began to shape his vision for what became his life's work—augmenting human intellect. The first version in the open literature was published in 1963 (Englebart, 1963).

By "augmenting human intellect" we mean increasing the capability of a man to approach a complex problem situation, to gain comprehension to suit his particular needs, and to derive solutions to problems . . .. We do not speak of isolated clever tricks that help in particular situations. We refer to a way of life in an integrated domain where hunches, cut-and-try, intangibles, and the human "feel for a situation" usefully coexist with powerful concepts, streamlined terminology and notation, sophisticated methods, and high-powered electronic aids. (Englebart, 1963, p. 1)

This is a clear statement of his vision; however, there is much more to say about Englebart's accomplishments. In 1965, Bob Taylor took over for Ivan Sutherland as Director of the Information Processing Techniques at ARPA and substantially increased Englebart's funding over that supplied by Air Force Office of Scientific Research. With the improved support, Englebart created the Augmentation Research Center at the Stanford Research Institute.

In 4 years, he made giant steps toward implementing his vision for a computer-supported environment for accomplishing intellectual work. Just as Lick had conceived of computers supporting an individual knowledge worker, Englebart surrounded the user with cathode-ray tube displays and a variety of experimental input–output devices beyond a keyboard. He wrote software for his ARPA-supplied, time-sharing computers that transformed them into a visionary prototype system for the support of intellectual collaboration called NLS, a somewhat forced acronym for oNLine System. He is credited with inventing the mouse, but in addition to the mouse and a standard keyboard, the environment included a five-key control box used to control information presentation. At the heart of NLS was a hierarchical information management capability that could deal with text, diagrams, pictures, even video conference imagery, could manipulate these objects symbolically, and display them in "windows" laid out on the multiple screens. He thought of text and pictures in hypertext terms. NLS could outline, expand, collapse, reorganize, and link paragraphs, illustrations, and sections hierarchically.

In the most dramatic computer demonstration of the 1960s, some would say the most dramatic ever, at the Fall Joint Computer Conference in 1968 in San Francisco, Englebart assembled all his gear on the stage. The tangle of equipment required to carry all this off might be characterized as kluggy, but the reader must appreciate the state of the early time-shared computers and displays he had to work with. He was using character generators and analog vector graphics, combined with television imagery in standard raster-scan formats. He piped the TV imagery from his Stanford laboratory and achieved switching among various hierarchical representations of these artifacts reasonably easily using his mouse and control box. He presented the results of 4 years work in 90 minutes to a large crowd who appreciatively gave him a standing ovation (Rheingold, 2000).

Englebart's most lasting contributions were the mouse, his working hypertext system as a part of NLS, and the vision for a personal workstation that can legitimately be thought of as one of the sources of ideas for the personal computer. He has continued his mission in life to provide computer support to the human intellect, but never with the funding and momentum of the NLS project. In 1989, he and his daughter, Christine, formed the Bootstrap Institute that continues to elaborate hypertext concepts well beyond what is found in current internet applications, and to consult with industry about how they can increase the efficiency and productivity of their knowledge workers.

Ted Nelson

The history of hypertext would not be complete without mention of the guru who invented the term, and typifies the 1960s and 1970s generation of hackers, Ted Nelson.

> Let me introduce the word *hypertext* to mean a body of written or pictorial material interconnected in such a way that it could not conveniently be presented or reproduced on paper. It may contain summaries, or maps, of its contents and their interrelations: it may contain annotations, additions and footnotes from scholars who have examined it. Let me suggest that such an object and system, properly designed and administered, could have great potential for education, increasing the student's range of choice, his sense of freedom, his motivation and his intellectual grasp. (Nelson, 1965)

Nelson writes copiously, with great vision, at times irreverently, but with great humanity, continuing to elaborate and refine his vision for international connectedness through hypertext (Nelson, 1965, 1973, 1974a, 1974b, 1981). He also tried, for more than 30 years, to create an uncompromised implementation of his vision, Xanadu. Although it never coalesced beyond beta versions, it has spawned a number of creative contributors to the field and served to focus Nelson's innovative thinking about hypertext.

Ivan Sutherland

While Licklider was talking and writing, Ivan Sutherland was producing code to describe his own visions. His MIT PhD thesis, Sketchpad, captured the attention of the fledgling computer science world immediately (Sutherland, 1963). The Sketchpad system not only opened the door to computer-aided design systems, but also became the basis for many of the graphics construction and editing capabilities that are commonplace today. Baecker and Buxton (1987) provide a nice summary of the far-reaching visions his code demonstrated:

1. The concept of a hierarchic structure to a computer-represented picture.
2. The concept of a master drawing and its instances, a forerunner of object-oriented programming.
3. The use of the method of constraints to specify the details of drawing geometry.
4. The ability to display and manipulate iconic representations of constraints.
5. The ability to copy, as well as "instance" both pictures and constraints.
6. Some elegant techniques for drawing construction using a light pen.
7. Separation of the coordinate system in which a drawing is defined from that on which it is displayed, permitting interactive three-dimensional manipulation of the drawing as an object, and recursive operations, such as move and delete, applied to hierarchically defined drawings.

The author personally remembers a remarkable early video from the project showing interactive rotation of three-dimensional objects.

It should be mentioned in passing on to the next epoch that the single-user features of the PDP-1 inevitably led to the development of the first computer game, Space War. It consisted of little more than missile-firing space vehicles with unidirectional thrusters in the inertial environment of space impacted only by the gravity of the sun. By orienting the vehicle and specifying the amount of thrust to be applied, each player moved about the screen and attempted to fire deadly missiles at each other. It started out as an MIT hack in the early 1960s, but was widely distributed to any organization that had the appropriate hardware. Little did the developers expect that such games would spawn a thriving industry.

1967–1977

State-of-the-Art

The period beginning in the late 1960s marked the end of the era of small independent machines and introduced the era of time-shared computing resources. With the impetus from Licklider and others, software and hardware infrastructures were introduced to make it possible to swap several individuals' programs and memory allocations between relatively fast magnetic core memory and relatively slow but much larger memory recorded on a magnetic drum so quickly that, when the machine was lightly loaded, users were barely aware that they did not have the full resources of the machine at their command. However, under heavy use, the machine began spending more time swapping users' work in and out than it did getting the actual users'

processing accomplished. The users began experiencing significant lags between when they typed and when the result appeared on their screen. This led to a flurry of HCI activity characterizing the taxonomy of system-induced time delays and research on how time delays affected user behavior. The seminal taxonomic paper, which also suggested acceptable delays for various conditions, was Miller (1968). Recently, it has seen a revival of interest directed toward web-based quality of service issues (Sears & Jocko, 2000). An example of the early user-delay interaction research is Morfield, Wiesen, Grossberg, and Yntema (1969).

Soon, geographically dispersed organizations with a frequent requirement for transaction processing, such as the insurance industry, the U.S. Department of Agriculture, the Social Security Administration, and other government organizations, became interested in taking advantage of time-sharing to implement "interactive" online transaction processing in the field and to connect their diverse locations to their central records repositories. The path was via leased telephone lines, through a series of regional data concentrators, and then to the central database machines. Only a few such systems were actually implemented. "Interactive" is in quotes because, by today's standards, there were usually substantial delays before a reply to an interactive query was received, and usually the central databases themselves were only updated overnight. These constraints launched the concept of the "screen" of information. The remotely connected systems, using a store-and-forward paradigm, would accumulate a "screen" or page of information and then transmit it to the central system via the concentrator. The concentrator would then download a new form or "screen" and set it up for response from the remote terminal. Initially, only the most advanced systems offered full duplex interaction, that is, interaction in which each ASCII character was transmitted and echoed by the host machine or concentrator as it is typed, rather than transmitted a line at a time or a "screen" at a time.

Human–Computer Interaction and Screen Design

In those days, HCI design was conceptually much easier than today, because the interface was defined anew for each application and was limited to a countable set of "screens" having a well-defined sequentially-branching structure. This was the state of HCI through much of the 1970s. Nevertheless, this stage of user requirements development, where the emphasis was on deciding what a user would want to see and what a user needed to type, led to much confusion and user frustration that was only occasionally addressed directly by usability studies. Martin (1973) and Galitz (1981) document the state-of-the-art of interface design during this time period.

Interactive Programming

Even more significant from the point of view of programmers' HCI was the development of interactive programming that replaced punched cards and seriously reduced the requirement for and the pain of batch compiling and code execution. Although programmers were already writing code at time-sharing consoles, the introduction of incremental compilers, infrastructure for writing programs in which each program statement was assembled incrementally, able to be verified in small sections and built up incrementally into a full application, made the process much more interactive and signaled the final end to the tedium and inefficiency of punched cards and batch program submissions. The first such programming language was JOSS, created at the RAND Corp. in the early 1960s (Marks & Armerding, 1967), but it was soon followed by Kemeny and Kurtz's BASIC (Kemeny & Kurtz, 1966) and BBN's TELCOMP (Myer, 1967).

Although it is hard to imagine today, the groups funding programming, in both commercial and government organizations, were skeptical at first that it was actually more cost-effective to program interactively than to use the batch systems. Since, it was argued, programmers sitting at interactive consoles would waste time trying different things, and the time delays of the then-current timesharing systems would inhibit timely results. Since, eventually, the entire program had to be tested as a whole anyway, why would it be more cost-effective to program interactively? Several HCI researchers were funded to evaluate the relative cost-effectiveness of the two approaches, for example, Gold (1969), Grant and Sackman (1967), and Sackman (1970).

As it turned out, the adoption of online programming techniques, most of which used incremental compilers, made punched cards and traditional batch submissions obsolete, regardless of the outcome of the experiments. In the 1970s, the first programming environments, such as INTERLISP and Smalltalk, were introduced. They included a whole range of programming support from integrated editors and debugging routines to property inheritance that supported the reuse of selective code segments. Following these developments, there was no longer even an argument that could be made supporting the batch mode of operation.

Human Factors Engineering

A brief diversion is needed to introduce the role of human factors engineering into this discussion. During the first half of the 20th century, the fields of industrial engineering and psychology both became involved in issues of engineering design to take account of human performance capacities and limitations. Industrial engineering came to it by way of Frederick W. Taylor's views of "scientific management" and time and motion study in industrial settings (Taylor, 1911). Psychology came to it by way of their involvement in personnel selection, training, and eventually design of military systems. The psychologists who entered this field, sometimes referred to as Engineering Psychologists, were interested in design to accommodate cognitive or intellectual capacities and limitations, as well as perceptual-motor skills. The Human Factors Society of America (now the Human Factors and Ergonomics Society) was founded in 1957 and had about 600 members by 1960 (5,000 members today). Because this group had always been interested in the design of systems to take account of their human users, it is only natural that they would become involved with the design of computers, software, and HCI more generally as the field emerged. In

fact, through the early 1970s, this group played the dominant role in introducing the concepts of cognitive psychology and psychological methodology into the design and evaluation of human–computer systems.

The notion that to begin man–machine system (sic) design, the engineer had to understand the tasks the system was required to perform, the context in which those tasks were required, and to make explicit the assignment of functions to various system components, including humans, was fundamental to the development of systems involving humans and machines. Furthermore, it became clear quite early that the programming of computers and the use of computers involved human cognitive and intellectual capacities and limitations that psychologists might be able to understand. Then, in the 1980s, when it came time to think seriously about the evaluation of user interfaces, the early efforts were based largely on experiment design methodology derived directly from experimental psychology. These facts account for the early and continued involvement of cognitive psychologists in the HCI field.

The first computer company to have a human factors group was IBM in their Federal Systems Division supporting military systems development during World War II. By 1956, there was a group in computer research at San Jose and soon thereafter at Yorktown Heights. However, at that time, IBM was largely a computer hardware manufacturer, and the research the group undertook was primarily concerned with factors that would influence how users would interact with their hardware, through control panels, keyboards, and hardware displays. It was not until 1979 that IBM, recognizing its larger role in software as well as hardware, undertook a serious initiative to build on human factors efforts already underway in a number of its labs, and introduced usability concepts and usability laboratories into its software design and development organizations (R. Hirsh, 2001, personal communication).

Alan Kay and the Palo Alto Research Center Nexus

During the 1970s, Alan Kay was carrying forward the visions initiated by Licklider, Englebart, and Sutherland. In fact, Kay got his start in computer science at the University of Utah in 1966 under Donald Evans and Ivan Sutherland. At Utah, he began dreaming of the idea of the personal computer. His first realization was embodied in his PhD thesis on the "Reactive Engine" (Kay, 1969). Also known as the FLEX machine, his software was truly innovative. It introduced some of the concepts of parallel processing, windows, and the message passing that is the foundation of object-oriented software. Kay was not interested in programming languages to support computation. He envisioned the computer as a tool for logical simulation and modeling of conceptual worlds. Accordingly, his object-oriented approach created structures whose complexity could be built up incrementally using a language that emphasized symbolic logic rather than computation. His vision, encouraged by his belief in Moore's Law,[3] was that we should eventually have a powerful, inexpensive, easy-to-use computer on everyone's desk.

In 1972, Kay moved from the Stanford AI Lab to join other innovators at the Xerox Palo Alto Research Center (PARC)—such as Butler Lampson, Bob Metcalf, Bill Newman, Charles Simonyi, Bill English, Chuck Thacker, Ed McCreight, Skip Ellis, and Bob Sproull—seeking to realize the dreams of Licklider and Englebart. The PARC was created in 1970, initially under the direction of George Pake (Waldrop, 2001). Here, Kay continued the evolution of the concepts in FLEX, which by now, with the collaboration of Adele Goldberg, had morphed into the Smalltalk programming system. Kay, influenced by the language, Simula, together with Goldberg, cast all computation in Smalltalk in the form of self-contained objects, each with its own interface, to the world that allowed partitioning into local and "public" information (S. Card, 2001, personal communication). As it evolved at PARC, Smalltalk became a programming environment in the sense that the programming language and the supporting software to facilitate development and debugging were seamlessly interconnected. These innovations, which are widely emulated today, were a giant step beyond incremental compilers toward object-oriented programming languages and interactive support for programmers.

At PARC, a team was developing the ALTO, the first truly powerful bit-mapped graphics personal machine (Lampson & Taft, 1979). The ALTO was a box the size of a two-drawer file cabinet and fit under the desk rather than on it, but it was clearly a step along the way toward Alan Kay's vision, for which by this time he had coined the term "Dynabook." It had a 256-kb core memory and a 2-megabyte removable hard disk required to support the object-oriented features of Smalltalk and INTERLISP. PARCers populated their offices with them and interconnected them through the PARC-developed Ethernet.

Information Processing Theory, GOMS, and Verbal Protocols

Now for a second digression. One significant stimulus for advancement in HCI and for the work at PARC came much earlier and from a somewhat unexpected source. In the early 1970s, Allan Newell and Herbert Simon began thinking quantitatively about the possibility of building an information processing theory of human problem solving and, more importantly, the possibility of embodying that theory in computer simulations of thought (Newell & Simon, 1972). This work may be considered to have spawned the whole field of AI, but for our purposes, it had two other salient impacts. First, to generate their theory, these authors began having the subjects of their experiments think aloud as they performed a problem-solving task and the resulting verbal protocols were considered to be data for their

[3]Moore's Law: The observation made in 1965 by Gordon Moore, cofounder of Intel, that the number of transistors per square inch on integrated circuits had doubled every year since the integrated circuit was invented. Moore predicted that this trend would continue for the foreseeable future. In subsequent years, the pace decreased a bit, but data density has doubled approximately every 18 months, and this is the current definition of Moore's Law. http://webopedia.internet.com/TERM/M/Moores_Law.html

theories (Ericsson & Simon, 1980). Simon and Newell's use of think-aloud protocols led HCI researchers to suggest that the *thinking-aloud method* could be useful for cognitive–interface design (Lewis, 1982). In fact, the method has, over the succeeding years, been widely adopted as an extremely profitable method for identifying the difficulties a user is having and pinpointing the causes of those difficulties and their remedies. (See Chapter 56, this Handbook.)

Second, in 1971, Allan Newell was consulting at PARC and proposed a project on applied information–processing psychology that would translate into practice some of his groundbreaking work with Simon. The project was begun in 1974 with the hiring of Stuart Card and Thomas Moran. Newell continued as a consultant. It is no wonder that with the generous financial support of the Xerox Corporation and the culture of the organization committed to the partnership between humans and computers that this HCI group became one of the premiere sites for ground-breaking HCI research.

They set as a goal to create a psychological theory that would have an influence on the ongoing technological developments of their colleagues. They thought it might be possible to use what was known in cognitive and perceptual psychology to create predictive models that could influence conceptual design and be used during the user interface design cycle to evaluate alternative designs before they were implemented (Card, Moran, & Newell, 1980a, 1980b). Their work culminated in the publication of *The Psychology of Human–Computer Interaction* (Card, Moran, & Newell, 1983). Not only did they develop useful predictive models, the book was also a tour-de-force that summarized all of the relevant human information processing literature in the behavioral sciences in the "Model Human Processor" and elaborated the keystroke model into a useful engineering application called GOMS (Goals, Objects, Methods, Selection Rules). It was a significant development that helped ground this emerging field in the early 1980s.

Human Sciences and Advanced Technology Centre and the British Community

Meanwhile, also in 1970, in Great Britain, Brian Shackel formed the Human Sciences and Advanced Technology Centre (HUSAT) at Loughborough University (Shackel, 1997). Although not exclusively interested in computer-related issues, under his leadership the Centre (now Institute) has played a leading role, along with T. R. G Green at the Computer-Based Learning Unit at the University of Leeds and the Applied Psychology Research Unit at Cambridge, during the tenure of D. E. Broadbent and later Alan Baddley, in applying the concerns, methods, and knowledge traditional to the fields of psychology and ergonomics (the term used in Europe for human factors engineering) to the study of computer design and use.

Design Guidelines and Style Guides

The period of the early 1970s, when time-shared remote terminal systems were becoming popular, was also the time when the first notions that there might be some user-oriented design principles that could be applied to the design of the "screens." Hanson is usually credited with the first statement of "User Engineering Principles" (Hanson, 1971). Hanson was an engineer at the Argonne National Laboratory who became interested in making Emily, a text editing system for programmers, as well adapted to his users as possible. In the course of constructing Emily, he derived four top-level principles:

- Know the user
- Minimize memorization
- Optimize operations
- Engineer for errors.

His paper then describes several subtopics designed to accomplish these higher level goals. Then, in 1975, Engle and Granda (1975), at IBM, using a variety of sources, assembled a much more comprehensive set of guidelines covering the topics:

- Display formats
- Frame content
- Command language
- Recovery procedures
- User entry techniques
- Response time requirements.

The development of guidelines generally evolved into the most comprehensive set developed by Sid Smith and his colleagues. His guidelines went through several revisions during the 1970s and early 1980s, culminating in the final version, Smith and Mosier (1986). This set of guidelines was so comprehensive that it was difficult to decide which specific recommendations apply to particular design requirements and to choose between two guidelines that in a specific context might suggest conflicting solutions for the same design detail.

As everyone now accepts, guidelines are limited to recommendations that can be applied without regard to the context of the specific application. This is a strength in the sense that they can apply broadly, but a weakness in that they are necessarily watered down to be general and unable to address issues that are conceptual rather than physically explicit, or that arise in a specific task context. Today, it is more common to generate style guides. A style guide usually is developed in reference to one product or product line for which the guidelines can be considerably more specific and contextually constrained. It often provides explicit examples of how the guidelines apply to the product(s) in question.

A very early style guide, perhaps the first, was developed by Pew and Rollins for the Department of Agriculture (Pew & Rollins, 1975; Pew, Rollins, & Williams, 1976). Although the system was never implemented, the Department was preparing to place interactive computers in their county field offices that would be connected to the central office databases in Washington, DC. They had a cadre of programmers who were experienced in producing software for batch operations, and the Department wished to retrain them to develop software

with which users, county agricultural agents, having no knowledge of how the computers worked, could interact and enter data as they conducted interviews with their farmer-clients. The management was concerned that the modules being developed by different teams needed to look the same to their users, who would be distributed all over the US. Accordingly, they commissioned the development of a Dialog Specification Procedure. In addition to providing specifications for what the screens would look like for different purposes, what terminology would be used, and how the user would interact with the systems, the Dialog Specification Procedures provided specific instructions on how to decide what information was required to be collected by the user in the field, that is, how to conduct task analyses, and specific procedures and templates for specifying the design of the screen layouts for the coders who would be implementing the programs.

Final 1970s Developments

As the 1970s were winding down, packet-switched networks were beginning to proliferate, and e-mail over such networks was just beginning to take hold with users in the research community. Frankston and Bricklin introduced the spreadsheet paradigm on an Apple II in the VisiCalc application in 1979 (Stranahan, 1996). It was a significant breakthrough because it blurred the line between the programmer and the user, making it possible for moderately sophisticated users to program simple repetitive mathematical and logical operations and to share the programs with other users. Finally, the first commercially successful video arcade game, PONG, produced by Atari, hit the market in several versions around 1973 (Winter, 1999).

1978–1988

State-of-the-Art

The beginning of this period ushered in a paradigm shift in the nature of computer interfaces. The approach to interface design went from a keyboard-controlled *glass teletype* running software having a frame-driven dialog imbedded in the application with rigid format and coding constraints to the fundamental features of the graphic user interface (GUI), well known today. Stimulated by the demonstrations of NLS, conceptions of the Dynabook, the Smalltalk object-oriented software programming environment, together with significant hardware improvements enabling much higher resolution bit-mapped displays and faster multiprocess hardware, the era of GUIs began. Bit-map graphics supported selective redisplay of screen images and sophisticated font and icon design. The improved hardware also included memory cashe systems that, in turn, supported multiple window displays.

The Xerox-developed display software, D-Lisp and the BRAVO text editor, implemented on the ALTO were the first instantiations (Teitelman, 1977) soon to be incorporated in the Xerox Star secretarial workstation in 1982 and then, of course, next in the Apple Lisa in 1983. and in the original Macintosh in 1984 (Bewley, Roberts, Schroit, & Verplank, 1983). No longer were abstruse command-line formatting instructions required in word processors, The text could be portrayed in literal page-layout format and printed without transformation on laser printers, another significant PARC development, so that "What you see is what you get!" (WYSIWYG). On the basis of quantitative human factors studies, Englebart's mouse was selected as the interactive device of choice to supplement the keyboard (Card, English, & Burr, 1978). The Xerox Star introduced the desktop metaphor that has become virtually a standard in today's GUIs. The concept of the Icon, also a Xerox invention, to represent desktop objects, was more than a simple pictogram. It provided a visual manifestation of the products of computing that had mnemonic value and could be buttoned, dragged, and dropped into other icons representing destinations as a means for carrying out intended actions on them (Bewley et al., 1983).

Perhaps even more importantly, facilitated by the object-oriented conception of software, the advent of the GUI brought with it the clean separation of the user interface code from the application code it was designed to support. Objects essentially had their own interface and made it possible to hide information internally that was not relevent to other objects. The user interface became a technology to be developed on its own. Ben Shneiderman coined the term, "Direct Manipulation Interfaces," to capture this paradigm shift (Shneiderman, 1983), but the term GUI has persisted in the literature.

On the one hand, the GUI made the computer much more accessible to nonprogrammer users who just wanted to do their work. On the other hand, it increased tremendously the sophistication of the things that a nonprogrammer user might want to do. On the one hand, it made the user interface much more intuitive. On the other hand, it increased the complexity and thereby increased the potential for user confusion and the challenge of user interface design because it relaxed many of the design constraints that previously existed. No longer could one layout a series of display frames in advance and be confident of all the actions a user could do next. No longer was one restricted to blinking and inverse video as text coding dimensions. The envelope of possibilities, manifest in multiple windows and applications, each with its own icons, menus, and widgets, seldom could be fully understood. Once again, the technological capabilities were outstripping the ability of the HCI community to accommodate user requirements.

Iterative System Development

It was during this time also that the industrial world was beginning to recognize the limitations of the waterfall system development or product development process and began experimenting with a more spiral process in which you build-a-little-test-a-little. This iterative approach had great appeal to software interface developers because it was much easier in software to develop prototypes that represented successive approximations to a final design. Furthermore, it created opportunities for the HCI and user communities to evaluate and critique designs in the formative stages before they were cast in stone. As early as 1986, Marilyn Mantei described how to incorporate typical