THE COMPUTER REACHES OUT: THE HISTORICAL CONTINUITY OF INTERFACE DESIGN

Jonathan Grudin

Aarhus University (on leave from MCC)
Computer Science Department
Ny Munkegade, Bygn. 540
8000 Aarhus C Denmark
(45) 86-127188 jgrudin@daimi.dk

ABSTRACT

This paper examines the evolution of the focus of user interface research and development from the first production of commercial computer systems in the 1950s through the present. The term "user interface" was not needed in the beginning, when most users were engineers and programmers; it may again become inappropriate when more applications are written for groups than for individuals. But there is a continuity to the outward movement of the computer's interface to its external environment, from hardware to software to increasingly higher-level cognitive capabilities and finally to social processes. As the focus shifts, the approaches to design and the skills required of practitioners changes. In this paper five foci or levels of development are identified. Most development today is positioned in the third level and considerable research is directed at the fourth. Some attention is now being given to the fifth: repositioning the interface in the work group or organization itself. Work at the different levels is not entirely independent, so establishing a comprehensive framework may enable us to position existing research and development efforts and plan future work more effectively.

INTRODUCTION

Ironically, "user interface" is a technology-centered term: the computer is assumed, the user must be specified.[1] And indeed, consideration of the history of that interface goes more smoothly if we position ourselves at a distance and think of the "computer interface" to the user and the world. This perspective afford us a single view that takes in the period before the term "user interface" was used and extends more gracefully into the future, when the computer will reach beyond individual users to understand and support groups and organizations.

When we consider only those interface techniques that are already in widespread use, the number of unresolved design questions is daunting. Unless one clones an existing product, designing even one aspect of an interface -- menu navigation, window operations, command names, function key assignments, mouse button syntax, icon design, etc. -- gives rise to a potentially endless series of decisions. The methods for arriving at informed choices are often too time-consuming and imprecise. Many more studies are needed if we are to develop an engineering base of appreciable utility. At the same time, user interface design is in a period of tremendous change. Color, graphics, sound, video, and animation are only beginning to be explored or widely applied. More sophisticated system foundations -- distributed, object-oriented, knowledge-based -- are just starting to find substantial markets. Other technologies are sure to appear. The field faces a major challenge in deciding where to invest its effort. This paper sets out a historical framework for understanding the options.

The principal focus of activity in computer development has moved gradually from hardware to software and is now shifting toward the user interface. Corresponding shifts are present within the domain of user interface research and development itself. We can plot the trajectory of work in human-computer interaction: the location of the "user interface" has been pushed farther and farther out from the computer itself, deeper into the user and the work environment. This in turn has led to new approaches to design and evaluation. And so it shall continue. We can extrapolate that new approaches, responding to the user interface's move into the workplace, will require new skills, supplementing current approaches. They may not graft easily -- or at all -- onto existing development practices. Already, accepted methods for developing good interfaces clash with efforts to standardize the development process; the approaches of the future will greatly amplify these problems.

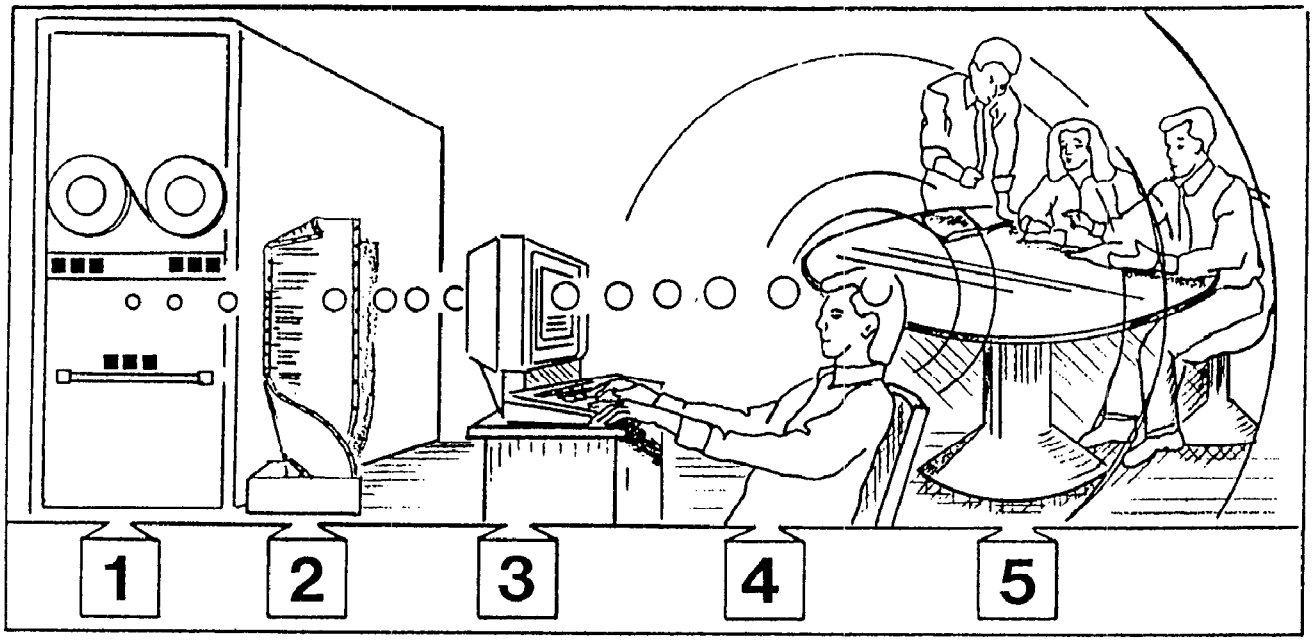

Figure 1. The five foci of interface development.

THE TRAJECTORY OF INTERFACE RESEARCH AND DEVELOPMENT

The Shifting Focus Of Computer Development

Twenty years ago, hardware remained the undisputed monarch of computer development. The major computer companies produced hardware and lived or died by its success. Simple processing benchmarks were the critical measure of new products. Microcoders were "soft" [16].

This changed in the late 1970s and early 1980s with the success of the spreadsheet, word processing, and licensed operating systems. True, this software primarily drove hardware sales, where the profits remained highest -- the major beneficiaries were IBM, Wang, and Digital, whose proprietary hardware was the hardware of choice for key software products in these areas.

Hardware innovation has a considerable future. Many computer companies still compete primarily at the level of hardware, accepting foreign competition and declining margins. But many companies that established themselves in the 1980s sell primarily software: Microsoft, Lotus, Ashton-Tate, and others. Debates go on within major companies over the wisdom of relying exclusively on profits from sales of "iron." For many companies, the business is changing. A manager of hardware engineering at a major computer company confided, "I wouldn't say this to my people, but a lot of hardware engineering these days consists of knowing how to use catalogs." The spread of workstations and standard platforms will extend the software focus that already exists in the PC world to more powerful machines. Software is moving to center stage.

The last five years have seen the beginning of the *next* step: a shift of marketplace attention to the user interface. The Macintosh interface produced profits first by driving the sale of proprietary hardware, but the myriad third-party Macintosh developers have used the interface to drive *software* sales. Software alone is profitable enough to justify a shift of attention to the user interface as a means of accelerating sales. This process is still at an early stage. The user interface draws more attention in mature software product areas. The appearance of new markets, where unadorned functionality often predominates, will slow the overall shift of focus toward the user interface. But the movement in that direction is inexorable.

The Shifting Focus Of Interface Development

Of course, systems have always had user interfaces: how have they evolved, prior to and since attracting attention? Again we find a series of changes in the focus of research and design, influenced by the changing backdrop of computer development. Figure 1 summarizes the shift in the principle focus of interface work. Initially, the user interface was located at the hardware itself -- most users were engineers working directly with the hardware. The focus then moved to the programming task -- higher-level programming languages and environments progressively freed users from the need to be familiar with the hardware. Next, with the widespread appearance of interactive systems and non-programming "end users," the user interface shifted to the display and keyboard, with early attention to perceptual and motor issues. Recent years have seen increasing research focus on the users' "conversational" dialogues with systems and applications, involving deeper cognitive issues underlying the learning and use of systems: the user interface is extending past the eyes and fingers, into the mind. Finally, with the advent of "groupware" and systems to support organizations, we are beginning to see the focus of user interface design extend out into the social and work environment, reaching even further from its origin at the heart of the computer.

The "outward" progression depicted in Figure 1 is natural. When we have solved the most pressing problems at one level -- *or can handle them adequately* -- human and computer resources are available to work on the next level. In a sense, the computer is colonizing its environment -- or, less threateningly, computers are progressively learning more about the world around them. This learning has generally been implicit, reflected in improved system and application design.

From this perspective, the "user interface" to the computer is misnamed, because it conjures up "a user" or a static cross-section of users. This obscures the continuity that extends through the two dramatic shifts in the user population: the shift from engineers and programmers to end users as the primary users, and the shift from supporting individuals to supporting groups that is just starting. That the latter shift can be perceived as a discontinuity is reflected in Malone's [17] coinage of the term "organizational interface" by analogy to "user interface." But if we remain in our own skin and consider the "computer interface" to users and the world, the discontinuities melt away and we see a smooth outward reach, as the computer "learns" about its environment.

1. The Interface At The Hardware

The first computer users were engineers who required a relatively full understanding of the hardware. They worked in binary, octal, or hexadecimal numbering systems, dealt directly with specific registers and memory locations, and whether engaged in scientific or business computing were focused on maximizing hardware performance.

The important point is that aspects of the hardware were dealt with by the *typical users* of the time, engineers and programmers -- hardware was a central part of the user interface. The user interface could be improved in three ways. First, more reliable and more capable hardware was developed. Second, the ergonomic presentation and manipulation of hardware components could be improved, for example by meaningfully arranging and labelling switches. Third, and most important, the user interface was improved by freeing the user, namely the programmer, from having to know about the hardware. This occurred through the development of higher-level programming languages, virtual memory, and so forth. This was not characterized as user interface development at the time, because when virtually all users were programmers the term "user interface" was not needed.

Even in the 1970s, the first home computers placed the user interface at the hardware [9] and more advanced computers still often brought users close to the hardware. Switch panels on computer exteriors provided direct manipulation of internal registers, although most users operated them in a cook-book fashion. Limitations such as physical memory size were more critical to programmer-users then than they are for most users of any stripe today.

Gradually, the distance between programmers (and other users) and hardware grew. The user interface was moving away from the hardware, away from the computer itself, beginning a journey that is still far from complete.

2. The Interface At The Programming Task

Through the 60s and mid-70s, computer programmers remained the principal users. Computer time and equipment were expensive. Early CRTs cost over $10,000. The first text editors were line-oriented editors designed for programmers; general use of computers for word processing was too expensive. In a true sense, improving the user interface meant improving programmer efficiency. The crucial advances -- still not characterized as user interface development -- were in higher-level programming languages, assemblers, compilers, debuggers, and operating systems. The new field of software engineering contributed concepts in virtual storage, data design, design methods, and software management that improved the programmer-user's interface to the computer.

Improving the ease of use for programmers was only one of several motivations underlying this work, which did *not* typically include human factors attention to the legibility of code, the memorability of terms, and so forth. Even in the case of programming languages, much of the focus was on providing access to higher-level abstractions and features that promote more structured programs [33]. Only toward the end of this period did a number of formal studies emerge of problem-solving in programming, debugging strategies, effective program documentation, and program visualization.

Emphasis on the programmer as user and computer programming as a central focus for user interface research was still evident at the 1982 Gaithersburg conference that led to the establishment of SIGCHI. Eight papers appeared in sections titled "Human Factors in Programming" and "Documenting and Developing Programs." The same focus was reflected in other papers as well; for example, "An analysis of line numbering strategies in text editors," [25] was one of eight papers on text editing; "Patterned prose for automatic specification generation," [28] was one of four on design guidelines.

The most significant boon to programmers was the development and use of multitasking, virtual memory, and interactive terminals. But interactive terminals soon changed the user interface even more dramatically by changing the user -- by creating vast non-programmer markets. The term "user interface" gained currency and a new field of research was established. Although attention to improving programming environments continues, emphasis within CHI has shifted from helping programmers-as-users to helping programmers develop better interfaces for non-programming end-users (e.g., papers on User Interface Management Systems outnumbered "new paradigms for programming" by about 3-to-1 at CHI'89).

3. The Interface At The Terminal

The visual display and interactive capability of terminals, home and computers, and workstations opened broad areas for development and research. Perceptual issues such as print legibility and motor issues such as comparative speed and accuracy arose in designing displays, keyboards and other input devices. Existing ergonomic or human factors work in these areas could be applied and extended. But the flexibility and extensibility of computers raised issues that went beyond the basic perceptual and motor processes, as function keys, command languages, menus, and other interface elements were developed. These created opportunities for cognitive psychologists to contribute in such areas as motor learning, concept formation, semantic memory, and action.

In a sense, this marked the emergence of the distinct discipline of human-computer interaction. It is illuminating to contrast the papers on computer interfaces at the 1981 Human Factors Society meeting with the papers at the Gaithersburg conference a few months later. They overlap in authorship, content, and heavy reliance on traditional experimental methodologies. But the human factors papers, scattered among papers unrelated to computing, focus heavily on sensory and perceptual aspects, while Gaithersburg drew numerous cognitive psychologists with broader, more theoretical interests.

Work on the perceptual and basic cognitive processes shared by most users remains the dominant focus of work in human-computer interaction. These issues were highlighted in Shneiderman's 1986 CHI plenary address on "central issues in human-computer interaction." [27]. More such work is needed as color, bit-mapped graphics, sound, larger displays, windows, and other capabilities become more widespread. These bring in graphic artists, with their different approaches to design and evaluation.

This ergonomic and cognitive research benefits all users of interactive terminals, but the focus of user interface research and development has shifted with the user population to "end users." And while the cognitive processing issues underlying menu and form layout, command name specificity, and other such design decisions are not fully resolved, some satisfaction is felt with emerging (and converging) graphic interface styles.

4. The Interface At The Interaction Dialogue

As the "level 3" perceptual-motor and cognitive results are being refined, extended to larger screens, incorporating color and sound, and responding to other technological advances, substantial "leading edge" research in human-computer interaction is taking on a deeper cognitive focus. This includes work on interfaces that develop a sense of the user by modeling users' goals or plans, by retaining past user actions to develop in the computer a sense of dialogue with the user, and by adapting or tailoring the interface to an individual over time [e.g., 13, 24]. In a metaphoric sense, the computer is extending its grasp beyond the keyboard and the display surface on which fonts, color patterns, and menus are arranged -- extending its knowledge into the mind of the user.

"Higher" functions studied in cognitive psychology and cognitive science are germane to this research. Human problem-solving, seen earlier in the studies of design and debugging by the programmer-as-user, is now an issue for more general users. Psycholinguistics can contribute to understanding dialogue. Studies of human planning and execution of complex tasks are relevant. With the weaker science base at this level, the dramatic arts may eventually contribute significantly [19, 20].

Technological advances that enable this development include the availability of memory and processing power, more sophisticated approaches to knowledge representation in software, and the spread of multi-tasking operating systems and software products that motivate a greater focus on dialogue and task organization.

Over the past few years, many experiments with user modeling, computer-based training and coaching, interactive advising, and adaptive systems have emerged. The research paradigms are not as well established as they are for studying lower cognitive processes and the methods are often less precise. Studies of planning and interaction dialogue rely less on controlled experiments measuring time and errors, and more on recording the dialogues and analyzing transcripts. This includes videotaping users' sessions, asking them to "think aloud," logging their keystrokes, and engaging in "Wizard of Oz" studies, where hidden experimenters test unimplemented capabilities by generating responses that the users believe are coming from the computer. Progress on the higher cognitive issues of level 4 is relatively slow, both because it is difficult and because the level 3 perceptual and basic cognitive processing issues remain the key concern of interface developers.

5. The Interface At The Work Setting

We can see increasing preparation for the next outward step of the interface, into the social or work setting. Since most work occurs in a social context, computers will support it more successfully if they implicitly or explicitly incorporate social and organizational knowledge. The spread of networked systems is of course a technological foundation for "groupware" or workgroup computing. Research and development areas include systems for electronic mail, co-authorship, distributed project management, and group decision support. However, progress in this area has been very slow.

Applications to support groups have all of the usual interface design problems and many more. Software supporting an entire group or organization will encounter individuals with a wide range of roles, skills, backgrounds, and preferences. Social, motivational, economic, and political factors arise that do not affect single-user applications. And studying groups is difficult -- group processes are often variable and context-sensitive, and usually unfold over time and in different locations; organizational change that results from introducing technology may take even longer to observe; and generalizing from observation is difficult -- each group's experience is governed by its constitution and the conditions under which technology is introduced [7].

	Level 1. Interface as hardware	Level 2. Interface as software	Level 3. Interface as terminal	Level 4. Interface as dialogue	Level 5. Interface as work setting
Principal users	Engineers/ programmers	Programmers	"End users"	"End users"	Groups of end users
Interface specialist disciplines	Electrical engineering	Computer science	Human factors, cognitive psych., graphic design	Cognitive psych., cognitive science, (dramatic arts?)	Social psych., anthropology, organizational...
Research methods	Largely informal	Largely informal	Laboratory experiment	Wizard of Oz, thinking aloud, data capture	Ethnographic, contextual, participant observer
Duration of basic events studied	Microseconds/ hours	Milliseconds/ hours	Seconds	Minutes	Days
Cost of evaluation	Lowest	Low	Moderate	High	Highest
Precision, generality	Highest	High	Moderate	Low	Lowest
Major focus	1950s	1960s-1970s	1970s-1990s	1980s-	1990s-

Table 1. Summary of the distinctions across levels of interface focus.

Factors contributing to the frequent failures of groupware include: i) it often requires that some people do additional work, people who are not the ones perceiving a direct benefit from its use; ii) it may lead to activity that violates social taboos, threatens political structures, or otherwise demotivates its users; iii) it may not allow for the wide range of exception handling and improvisation that characterizes much group activity; iv) developers' intuitions are especially poor for multi-user applications; v) we fail to learn from experience because of the difficulty of generalizable analysis and evaluation [10].

Formal studies of group settings often originate in business or management schools. Relevant techniques include those of social psychology [31] and anthropology [e.g. 23, 29, 34]. Approaches that have been used include "contextual research," in which users are interviewed in their work contexts [32], "participant observer" studies, in which a researcher joins and participates in the activity of a group [22], and "participatory design," in which system designers form a full partnership with the eventual users of a system [3, 4]. In addition, technology has been placed in a broader setting by examining the interplay over time of individuals and artifacts [5, 14, 18].

SUMMARY

The first row of Table 1 lists the principal computer users for each level of interface focus. The second row lists specialized disciplines addressing interface issues at that level. Of course, most actual interface research and development has been the work of programmers and software engineers.

Next are methods employed by these specialists. Interface improvements for engineers and programmers have generally been designed and evaluated informally. Laboratory experiments dominated the human factors and early cognitive psychology work on low-level processes. Protocol collection and analysis are central to studies of dialogue. Methods largely drawn from the social sciences are being used to examine technology in the context of the workplace.

The fourth row lists the duration of the basic events that concern the interface designer. The programmer interface drew attention both to hardware and software events, measured in fractions of seconds, and to improving the efficiency of programming, measured in programmer hours. The basic perceptual and motor events measured in level 3, even in studies of learning, have generally been on the order of seconds. Interaction dialogues typically cover substantial segments of a user's session with an application or system and may span multiple sessions. Finally, much longer durations are involved in studies of the work setting. A simple group process such as sending and receiving an electronic mail message may involve a considerable physical and temporal span, and many social processes unfold over weeks or months.

The next two rows address the difficulty of evaluation, as measured in the cost of running a test and the ease of generalizing from its results. It is usually unnecessary or straightforward to test whether improved hardware benefits users. Similarly, while programmers do not always agree among themselves, it is relatively easy to compare the time taken to code or modify a routine or application in different environments. (Of course, coding time is only one factor considered in choosing a development environment; tradeoffs always accompany usability issues.). Results in these areas generalize relatively broadly. This picture changes as end user efficiency becomes the object of design. A tremendous number of issues and alternatives arise, making design and evaluation expensive; users vary widely, making generalization difficult; and the science base is not as well

understood [e.g., 2]. The problems increase as we move to the higher-level processes governing dialogue; they are less well understood and their longer time course increases the methodological uncertainty and the cost of study. Finally, group processes magnify these difficulties substantially, with long time courses, great differences in individual and group composition, little science base, and huge effects on performance resulting from a host of local conditions [7, 10].

The final row contains approximate intervals during which each user interface level was in substantial focus. Improving hardware was the dominant concern for the first commercial computers of the 1950s. The 1960s and early 1970s were the focus for the analysis and exploration of higher-level programming languages [32] and for the other developments that progressively freed programmers from the hardware, including multitasking and virtual storage [15]. The advances of those periods have been consolidated and extended, but since then user interface attention has shifted from programmers as users to end users. A strong focus on basic perceptual, motor, and cognitive functions began in the mid-to-late 70s, leading to the formation of SIGCHI and other organizations and conferences. This concern with "look and feel" continued through the 1980s and is converging with the development of several similar user interface standards. It will remain in focus through the 90s as other modalities and capabilities are widely adopted. The 1980s have seen exploratory research on higher-level cognitive aspects of interfaces; this work is likely to move out of research laboratories and into development projects in the coming decades. Finally, the work setting as a determinant of user interface design for groups is just coming into view in the United States and seems destined to gain in research prominence in the 1990s.[2]

Each level of user interface focus has its own practitioners. Each relies on a science base that is less mature than the previous level, produces results more slowly and of less generality than the previous level, and therefore will take longer to work through. In a sense, we are teaching the computer about ourselves and the world, and in areas where our knowledge is less complete, it takes longer.

A DEVELOPMENTAL ANALOGY

The development path has some similarity to a growing child's. The computer first shared an infant's focus on basic physical functions (hardware), then developed conscious (software) control over the limbs (peripheral devices) that support the basic functions. It next expanded its perceptual and motor relationship to the outside world, followed by cognitive development, and finally a concern with social relationships and structures. These advances accompanied the computer's growth -- in power and quickness, though not size! And as with the computer, each successive "stage" requires more time and effort to achieve proficiency. Extending this half-serious personification, the computer-infant initially interacted exclusively with those who fed and healed it -- engineers as parents, happy to learn the halting language and primitive thought patterns of their charge, but nevertheless very pleased when it was able to take control of its own "basic physical functions." Next came those who would still learn its language and adapt to it to a great extent, while trying to educate it -- programmers as teachers. Now it is reaching out to individuals who are less inclined to adapt to it -- end users as community members. And finally will come social understanding, where the onus may shift even more to the computer to be cognizant of and considerate of its environment. In a manner somewhat similar to a growing child, the computer is reaching out into its environment.

INTERDEPENDENCIES AMONG THE LEVELS

The analogy extends to another important point. Psychologists have used the concept of "stages of development" to reflect the observation that children acquire different skills at different points in time. Yet this has been found to be at best an approximation to what happens [6]. Activity in different "stages" overlaps, and changes in one skill can affect performance in the others. Progress at higher levels may be restricted by incomplete knowledge at earlier ones, but mastery at one level is not necessary for attention to be directed to others. For example, although our social awareness may be limited or influenced by the maturity of our skill at more fundamental reasoning and planning, social and cultural learning begins early and influences cognitive processing at all levels. Even the underlying "hardware," the brain itself, continues to mature through childhood (and changes throughout life), with consequences for learning and reasoning at all levels.

The same is true with interface development. We can identify "stages" during which the user interface at the hardware and then the interface at the software received the most attention. More recently, most interface development has been focused on the perceptual and cognitive issues of "level 3." However, even in the 1950s, a few researchers anticipated level 3 issues [26]. And today, exploration of hardware and software "user interface" issues continues; for example, greater hardware reliability and object-oriented programming techniques have significant bearing on work in levels 3, 4, and 5.

In the present, there is a partial polarization of the field, with most development in the final period of level 3 work and many leading research organizations exploring level 4 topics. This may partially account for the mixed reviews given paper sessions at CHI Conferences -- a mismatch between the interests of researchers and developers that arises because CHI naturally wants to capture the leading edge of research. It is important though for level 4 researchers to appreciate that only when details are better worked out at level 3 will developers look to level 4 research for the next way to improve and differentiate their products, and equally important for level 3 developers and researchers to recognize that this period is ending and that the level 4 issues, which may seem foreign or uninteresting now, may soon be much more important.

In fact, because of the influences that work across levels, *optimal* solutions for level 3 design issues actually

require information about how individuals and groups work over time that is acquired through level 4 and level 5 approaches. For example, the optimal design of features such as function key placement, command name abbreviation, and menu defaulting requires specific knowledge about the users' work practices and environment [11]. Refining level 3 designs may require the research and development techniques here associated with levels 4 and 5.

Looking ahead, the emergence of numerous quite similar graphic interfaces signals that the level 3 issues are being resolved through standardization on an adequate (though not necessarily optimal) set of features and behaviors. This will free many researchers and developers to work elsewhere. But the move to levels 4 and 5 will be slowed by the emergence of new modalities requiring perceptual and basic cognitive analysis (sound, video, animation...) and by the time needed to acquire interest in the new concepts and skill with the new techniques of levels 4 and 5 or to recognize the need to hire people who have them.

In the meantime, there will be disagreement over the best allocation of existing resources: software environment improvements to accelerate change, perceptual and cognitive studies that translate relatively directly into engineering, cognitive theory that underlies higher-level dialogue, or social and organizational research that places technology and cognitive processes in a broader work context.[3] In particular, one might question the need to address level 5 issues at all, given their difficulty, our lack of familiarity with the appropriate techniques for designing systems for groups, and our need for more work at other levels. The software development world is just beginning to accept human factors work on level 3 issues, and is unlikely to adjust easily to the greater cost, duration, and uncertainty associated with the level 4 and especially level 5 techniques. Software development organizations and the development process as widely practiced today were structured without the special needs of user interface design in mind [12], and already the conflict between the desire to routinize software development and the uncertainties of user interface design make it difficult to employ widely endorsed methods for user interface development [22].

However, as noted above, optimization at other levels often requires "looking out" into the workplace. In addition, in mature application areas, where products with similar functionality are converging on similar "look and feel," vendors will seek new grounds for distinguishing their product. Finally, as with the child, mastery at one level is not required to progress in another. In fact, once a child is interacting with other people, you might wish that the child's higher reasoning powers were more developed, but you had better start teaching the kid some basic social skills right away! The spread of networked computers and computer use in general means that, for better or worse, the computer *is* reaching out into groups and organizations, where the potential for subtle problems is high. Anything that increases the computer's effectiveness will be welcome. We need work at all levels.

POSTSCRIPT: THE ENGINE OF CHANGE

J. Robert Oppenheimer [21] said that an analogy is most interesting where it breaks down. The metaphor of the computer as child reaching out into the world is offered as a stimulus for thought, but has limitations. One is that whereas children master *their own* physical, perceptual, cognitive, and social functions, this paper presents the computer as gaining control over its own hardware and software functions, but then increasing its knowledge of *our* perceptual, cognitive, and social behavior. It does not address the computer's perceptual growth, reflected in scene or speech recognition, for example, nor does it address advances in machine reasoning. This could be done -- it would be another paper and would be truer to the metaphor of the child. I chose not to do so because I feel that we have discovered that these abilities of the computer are less important than its ability to "understand" us. This reflects the complete subservience of the computer to our collective will. The computer has a great stake in understanding how we think and act. Of course, the computer is not reaching out only into our minds and organizations, it is also reaching out into the world, as new application areas spring up and as existing application domains are developed in ever greater depth. This is no surprise, since computers are not marketed for the purpose of understanding their users -- they are built to act on the world. But they are built to act on the world *for us*, and to communicate the results of their actions *to us*. Their capability for autonomous action will always be sharply limited by their ability to understand our needs and communicate with us. Their effectiveness as agents in the world will increase in step with their greater understanding of us. For that reason, work to develop an understanding of people will remain at the very heart of the computer's development. It is the engine of change.

FOOTNOTES

[1] Bannon [1] has noted the pervasiveness of the linguistic bias in the human-computer interaction field. "Casual users" is a term often used to describe managers and executives -- who are often not at all "casual"! "Novice" or "naive" users are often expert at their jobs -- while the expertise of "expert users" may not extend beyond computer use. Our terminology simply assumes that everything is in reference to the computer. Other terms that have been used include "non-professional user," "non-specialist user," and "idiot-proof programs" [8]. The early 1980s saw some resistance to the term "user interface" because many non-technical people associated "user" with "drug user." Non-technical people, if asked for the term that best matches a description of the human-computer interface, would probably prefer "computer interface" to "user interface."

[2] In their ambitious framework, Gaines and Shaw [8] date some of the advances mentioned here about a decade earlier. They generally mark change by its first arrival in a few research labs, whereas I have tried to identify when the use of a technology became reasonably widespread in the computing world.

[3] The interface at the hardware is omitted from this list because, as noted above, the era of hardware dominance in computer development is ending. Although hardware improvements will continue to have a major impact, we are singularly unable to exploit fully our existing hardware.

ACKNOWLEDGMENT

Jim Hollan encouraged me to organize these thoughts. Seeing my "level 4" colleagues at MCC and elsewhere labor to position their work in the field helped me realize that we are all working usefully toward a common goal. Bob Glushko, Karen Holtzblatt, and Bill Kuhlman commented usefully on earlier drafts. Phil Barnard, Robert Mack, Dave Wroblewski, and the Aarhus IMV group (with Liam Bannon) made especially helpful suggestions.

REFERENCES

1. Bannon, L., 1990. From human factors to human actors. In J. Greenbaum and M. Kyng (Eds.), *Design at work.* Hillsdale, NJ: Lawrence Erlbaum Associates
2. Barnard, P. and Grudin, J., 1988. Command names. In M. Helander (Ed.), *Handbook of Human-Computer Interaction.* Amsterdam: North-Holland.
3. Bjerknes, G., Ehn, P., and Kyng, M. (Eds.), 1987. *Computers and democracy - a Scandinavian challenge.* Aldershot, UK: Gower.
4. Bødker, S., Ehn, P., Knudsen, J., Kyng, M., and Madsen, K., 1988. Computer support for cooperative design. In *Proc. CSCW'88 Conference on Computer-Supported Cooperative Work.*
5. Carroll, J.M. and Kellogg, W.A., 1989. Artifact as theory-nexus: Hermeneutics meets theory-based design. In *Proc. CHI'89 Human Factors in Computing Systems*, (Austin, April 30 - May 4).
6. Cole, M. and Cole, S., 1989. *The development of children.* San Francisco: W.H. Freeman.
7. Ehrlich, S.F., 1987. Strategies for encouraging successful adoption of office communication systems. *ACM TOOIS, 5,* 340-357.
8. Gaines, B.R. and Shaw, M.L.G., 1986. From timesharing to the sixth generation: the development of human-computer interaction. Part 1. *Int. J. Man-Machine Studies, 24,* 1-27.
9. Gentner, D.R. and Grudin, J., 1990. Why good engineers (sometimes) create bad interfaces. In *Proc. CHI'90 Human Factors in Computing Systems.*
10. Grudin, J., 1989. Why groupware applications fail: Problems in design and evaluation. *Office: Technology and People, 4,* 3, 245-264.
11. Grudin, J., 1989. The case against user interface consistency. *Communications of the ACM, 32* (October), 1164-1173.
12. Grudin, J., 1990. Organizational obstacles to participatory design in large product development organizations. Unpublished manuscript.
13. Hollan, J., Miller, J.R., Rich, E., and Wilner, W., 1990. Knowledge bases and tools for building integrated multimedia intelligent interfaces. In Sullivan, J.W. and Tyler, S.W., *Architectures for intelligent interfaces: Elements and prototypes.* Reading, MA: Addison-Wesley, in press.
14. Hutchins, E., 1985. The social organization of distributed cognition. Unpublished paper.
15. Jacobs, S.M., 1984. Operating systems. In Vick, C.R. and Ramamoorthy, C.V., (Eds.), *Handbook of software engineering.* NY: Van Nostrand Reinhold.
16. Kidder, T., 1981. *The soul of a new machine.* Boston: Little, Brown.
17. Malone, T.W., 1985. Designing organizational interfaces. In *Proc. CHI '85 Human Factors in Computing Systems*, (San Francisco, April 14-18).
18. McKendree, J. and Mateer, J. W., 1989. Film techniques applied to the design and use of intelligent systems. MCC Technical Report ACT-HI-261-89.
19. Mountford, J. (Moderator), 1989. Drama and personality in user interface design. In *Proc. CHI'89 Human Factors in Computing Systems.*
20. Norman, D.A., 1990. Cognitive artifacts. In J.M. Carroll (Ed.), *Theory and design in human-computer interaction,* in press.
21. Oppenheimer, R., 1956. Analogy in science. *American Psychologist, 11,* 127-135.
22. Poltrock, S., 1989. Innovation in user interface development: obstacles and opportunities. In *Proc. CHI'89 Human Factors in Computing Systems.*
23. Reder, S., and Schwab, R.G., 1988. The communicative economy of the workgroup: Multi-channel genres of communication. In *Proc. CSCW'88 Conference on Computer-Supported Cooperative Work*, (Portland, September 26-28).
24. Rouse, W.B., Geddes, N.D., and Curry, R.E., 1987-1988. An architecture for intelligent interfaces: Outline of an approach to supporting operators of complex systems. *Human-Computer Interaction, 3,* 87-122.
25. Schneider, M., Nudelman, S., and Hirsh-Pasek, K., 1982. An analysis of line numbering strategies in text editors. In Proc. *Human Factors in Computing Systems*, Gaithersburg, MD.
26. Shackel, B., 1959. Ergonomics for a computer. *Design, 120,* 36-39.
27. Shneiderman, B., 1986. Seven plus or minus two central issues in human-computer interaction. In *Proc. CHI'86 Human Factors in Computing Systems*
28. Smith S., 1982. Patterned prose for automatic specification generation. In Proc. *Human Factors in Computing Systems*, Gaithersburg, MD.
29. Suchman, L., 1983. Office procedures as practical action: Models of work and system design. *ACM Trans. on Office Information Systems, 1,* 320-328.
30. Suchman, L., 1987. *Plans and situated actions: The problem of human-machine communication.* Cambridge: Cambridge University Press.
31. Vaske, J.J. and Grantham, C.E., 1989. *Socializing the human-computer environment.* Norwood: Ablex.
32. Wegner, P., 1976. Programming languages -- the first 25 years. *IEEE Trans. on Computers, C-25, 12, 1207-1225.*
33. Wexelblat, R.L. (Ed.), 1978. Proceedings of ACM SIGPLAN History of Programming Languages Conference. *SIGPLAN Notices, 13,* 8.
34. Whiteside, J., Bennett, J., and Holtzblatt, K., 1988. Usability engineering: our experience and evolution. In M. Helander (Ed.), *Handbook of human-computer interaction.* Amsterdam: North-Holland.