RC 13926 (#62583) 8/16/88
Human Computer Interaction 21 pages

Research Report

Evaluation, Description and Invention: Paradigms for Human-Computer Interaction

John M. Carroll

User Interface Institute
IBM T.J. Watson Research Center
Box 704
Yorktown Heights, New York 10598

BOOK or CHAPTER

IBM Research Division
Yorktown Heights, New York • San Jose, California • Zurich, Switzerland

Evaluation, Description and Invention: Paradigms for Human-Computer Interaction

John M. Carroll

User Interface Institute
IBM T.J. Watson Research Center
Box 704
Yorktown Heights, New York 10598

Human-computer interaction (HCI) is an urgent and rapidly developing area of computer science research and application. As it continues to evolve and to define itself, it is possible to identify distinct paradigms, or orientations to HCI research and application. Initially, HCI work *focussed on empirical* laboratory *evaluation* of computer systems and techniques. Subsequently, empirical studies of usability were organized by and addressed to cognitive theoretical *description* of user behavior and experience. Currently, the focus of HCI work is shifting toward a more directive role in *invention*, design and development of systems and techniques. The progression of these three paradigms comprises a case study of a field discovering what it is about, and more generally, of the variety of roles available in the psychology of technology.

To appear in M.C. Yovits (Ed.) **Advances in Computers, Volume 28.**
New York: Academic Press, 1989.

CONTENTS

A vivid image of the recent evolution of computer technology is that of a "race" between function and usability. New technologies, new capabilities become available to users faster than user problems can be studied, understood and addressed. For example, the many user studies of word processing applications carried out over the past decade focused their attention on keyboard oriented, stand-alone systems with small and low-resolution monochrome displays. In 1981, our group at the Watson Research Center turned attention to secretaries learning to use such word processing applications. At the time, this was a novel application; computer editing was still largely the province of programmers revising code.

But now, and without a finished analysis of word processing, the frontier of usability has been pressed onward by the development and introduction of new applications and new interface technologies. Communication applications such as electronic mail and computer conference support raise usability challenges far more diverse than those raised by the extension of word processing to nonprogrammers. In the current technology, multiple users cooperatively access multiple applications via an extremely heterogeneous collection of workstation types. And even as the usability issues in these new domains are being articulated and explored, leading-edge prototypes are introducing gestural (e.g., handwriting) and speech input and interactive video output. Such new developments are occurring more rapidly, more broadly across the industry, and impacting more users all the time.

The race between function and usability has made the area of human-computer interaction (or HCI) a very high-profile research area within computer science and within the computer industry: it is difficult to develop usability science and technology fast enough, but it is also critical to do so. Indeed, the race has created the need for chapters like this one. However, this attention has also helped to expose some fundamental perplexity about what the field is and how it is supposed to work. It is still the case that HCI research has its principal effect on *discussions* of usability and user interface design and only a small, derived effect on actual *practice* in the design and development of computer systems and applications.

What is the goal of HCI research? There need not be a single answer to this question. But the more answers there are, and the more irreconcilable the various answers are, the more fragmented the field will appear. In HCI there are many answers to this question. One traditional answer comes from the field of Human Factors: HCI needs to provide methods and metrics for evaluating the usability of computers. A second answer comes from Cognitive Science: HCI is a testbed for the application of cognitive psychology to a real problem domain. A third answer comes from the exigencies of the computing industry: HCI must help guide the definition, invention and introduction of new computing tools and environments.

The practice of HCI is even more fragmented than its goals might imply. For example, some varieties of human factors evaluation explicitly suggest that developing cognitive science theories of HCI may *impair* progress in understanding usability (Whiteside and Wixon, 1987). On the other hand, Newell and Card (1985) warn that psychology might be driven out of HCI by computer science unless it can develop predictive cognitive models, coining the slogan "hard science drives out the soft." Yet even the most developed cognitive models in HCI have had no significant impact on the design of user interfaces (Carroll and Campbell, 1986). Moreover, it is paradoxically true that product innovations in user interface design have generally *led* HCI research rather than following from it in the conventionally assumed flow of "technology transfer" from Research to Development. The recent impact of the Apple MacIntosh illustrates this.

Perhaps these conflicting and fragmented views of HCI can be understood as consequences of the race between function and usability, of the rapid growth in needs, activities and expectations. Perhaps the current perplexity about HCI reflects an intermediate state in a true evolution toward more effective approaches to understanding the usability of computer systems and applications. In this chapter I take such an historical view, identifying three distinct paradigms, or orientations to HCI research and application. Initially, HCI work focussed on empirical laboratory *evaluation* of computer systems and techniques. Subsequently, empirical studies of usability were organized by and addressed to cognitive theoretical *description* of human behavior and experience. Currently, the focus of HCI work is shifting toward a more directive role in *invention*, design and development. The progression of these three paradigms comprises a

case study of a field discovering what it is about, and more generally, of the variety of roles available in the psychology of technology.

1. Human Factors Evaluation

The traditional role of psychologists working in the context of computer applications and services is empirical evaluation of usability. The original research arena of human-computer interaction is the psychology of programming and the professional programmer (Curtis, 1985; Shneiderman, 1980). A prototypical example of this paradigm is a set of experiments conducted by Sheppard, Curtis, Millman and Love (1979). In one of these, participants were given 20 minutes to reconstruct from memory a Fortran program of 26-57 lines that they had studied for the preceding 25 minutes. Two approaches to "structured" program organization (linear sequence, structured selection and structured iteration; Dijkstra, 1972) were contrasted with a "convoluted" organization (including backward exits from DO loops, arithmetic IFs, and unrestricted GOTOs). Reconstructive memory for the convoluted program organization was poorer (i.e., error rates were higher) than for either of the structured organizations (though only in one case was the difference statistically significant).

Such early work in the human factors of programming was important in demonstrating the feasibility of empirical assessment. By addressing some of the timely issues of the day, it broadened the grounds of debate in software technology from formal analysis and system performance to include usability and productivity issues. The basic paradigm of directly comparing two alternate designs in a usability evaluation is still the standard of practice in much HCI research and in many product development laboratories.

1.1 Direct empirical contrast

The development of empirical methodologies for evaluation, and the exercise of these methodologies in the context of software and system design, is a continuing need in HCI. Direct empirical measurement is still the only adequate means of assessing the usability of software techniques and computing artifacts (Carroll and Rosson, 1985; Curtis, 1980; Gould and Lewis, 1985). Establishing the importance of usability to the success of computing systems and techniques, and developing and promoting empirical methodologies to make usability evaluations have been major foci of HCI work.

From the start, HCI evaluation studies were strongly influenced by research practice in experimental psychology: emphasis was placed on tightly controlled laboratory approaches. From an historical standpoint, this was a reasonable move: there was an acute lack of theory and methodology for investigating usability. These laboratory studies generally took the form of direct contrasts: computing artifacts or techniques were directly pitted against one another in a brief but behavior-intensive measurement session. This evaluation work produced a variety of findings, often framed as guidelines for software development practice and user interface design, generally of the form "A is better than B." And perhaps even more importantly, the work set a more objective standard for usability evaluations, and provided a systematic basis for scrutinizing designers' hopeful intentions and trade press reviewers' glib comments.

However, there are many limitations inherent in the laboratory-based direct contrast methodologies of experimental psychology. These limitations became clear when the methodologies were applied in the complex practical contexts of HCI design. Controlled laboratory studies of software are difficult to design and carry out. The investigator needs to master programming languages and computer applications in order to be in a position to assess others' performance and to interpret their experiences. The experimental tasks that are studied necessarily require skilled human participants and involve learning and using very complex tools. This is expensive and time-consuming research. Such difficulties just don't come up when one takes an experimental approach to memorizing nonsense syllables, the stock-in-trade of traditional experimental psychology, or to making timed responses to meaningful but simple objects like isolated words, its more modern variant.

In experimental psychology, the sheer differences in recall rate or response times may be all there is to know about a person's performance in a task: the situations are relatively simple.

Understandably perhaps, such work is directed at collecting straightforward quantitative indicators of performance like task times and error rates, and formally testing these for statistical significance of direct contrasts (that is, computing the probability that obtained score differences might have occurred by chance). HCI situations, however, are not simple at all. In many cases it may be more important to know how people approach a task, or how they feel about their performance, than it is to know how quickly or successfully they perform. Nevertheless, the early commitment of HCI evaluation work to direct contrast studies created a strong bias for collecting quantitative indicators of performance, like time and success measures, and against placing primary, or even equal emphasis on qualitative data (which in other human factors contexts have often played a more prominent role; Chapanis, 1959: 23-95).

These constraints of direct contrast laboratory methods took a toll on the relevance of HCI evaluation work. The difficulties of designing and conducting controlled experiments in complex circumstances, inclined investigators to make use of scaled-down tasks, for example, memorization and reconstruction of small programs. The focus on quantitative differences inclined investigators to focus on the simplest of performance measures. This undermined the fundamental objectives of human factors evaluation, transforming questions about complex human behavior and experience in complex computing environments into simple scores of performance on toy-scale tasks. Such work could not answer the underlying *"why" questions* that motivated human factors evaluation in the first place; it could not provide the depth of understanding necessary to help guide the design of new software techniques and applications.

Yet this style of work became quite pervasive. Ledgard, Whiteside, Singer and Seymour (1980) assessed the use of symbolic notations in text editor commands by contrasting a command language having extremely complicated symbolic conventions with one almost free of these. Murrel (1983) contrasted message-based and window-based communication for a cooperative decision-making task. Holt, Boehm-Davis and Schultz (1987) contrasted object-oriented design with more standard approaches. But exactly what is it about symbolic notations that is bad? What is it about window-based communication and object oriented design that is good? None of these projects resolved the over general evaluation issue it posed. And none collected detailed enough information to contribute to a conceptual understanding of the issues involved.

Worst of all perhaps, these simplifications frequently did not even produce the statistically significant differences they were adopted to facilitate. The use of indentation to highlight structure in program listings seems intuitively like a good idea. It's a simple factor that can in principle be conveniently removed from the complications of the real programming process for direct contrast laboratory study. However, Love (1977), Shneiderman and McKay (1976) and Weissman (1974) all failed to find significant benefits of indentation. Studies of variable names have produced a conflicting potpourri of results; sometimes mnemonic names are more effective than non-mnemonic names and sometimes not (Shneiderman, 1980: 70-71). The daunting possibility remains that it was *because* of the trivial tasks that were studied and the limited types of data that were collected and analyzed that no differential benefits were found.

Such practical problems with direct contrasts encouraged experimental designs contrasting extreme positions, again to increase the possibility of measuring statistically significant differences. Ledgard et al.'s (1980) assessment of symbolic conventions contrasted extremely complicated examples of such conventions with an extreme absence of them. Liebelt, McDonald, Stone and Karat (1982) showed that a menu system was easier to learn when the menu hierarchy was organized than when it was disorganized (!). Indeed, in the Sheppard et al. (1979) experiment, several alternate approaches to "structured" programming were consistently indistinguishable based on the data, however the extreme alternative of "convoluted" programming produced significantly poorer performance than one of the structured approaches. In a sense, this study did not so much verify the benefits of deliberately structuring code as it did the risks of deliberately mis-structuring it. (Obvious and extreme evaluation contrasts are still sometimes professionally encouraged as long as they employ "an interesting methodology," Green, 1987: 6.)

Finally, human factors evaluation work is highly constrained by the often prodigious amounts of time required to make direct experimental contrasts of alternatives. Indeed, it seems logically doomed to consume more time than the evolution of software it is intended to guide. By the time the Sheppard et al. (1979) paper appeared, structured programming methods were

already the established practice. The evaluation work confirmed what had already happened, rather than playing a causal role in the evolution of practice. This limitation of the evaluation paradigm for HCI could be called the "evaluation dilemma": one cannot evaluate something that does not yet exist, hence direct evaluation always lags development by some fraction of a development cycle (Carroll, 1987a).

In sum, the exigencies of direct contrast laboratory work entrained compromises in the face validity of the work itself, and in the end, often failed to produce definitive or timely evaluations. How should programs be structured? How should hypertextual information systems be navigated? One cannot answer these questions with a few simple performance measures, but they are surely *empirical* questions. Answering them would involve developing a detailed understanding of what people do and try to do with programs and applications and the rich interaction of these goals and actions with the constructs of programming languages, the facilities of computing environments, aspects of the workplace, and many other factors.

These complexities have had a predictable effect: even in quarters where human factors evaluation is the official operating paradigm, most of the impact of psychology on the development of technology has come about through task analysis or consulting. Indeed, to a considerable extent human factors evaluation has become an historical stage in the development of current HCI. We return to the curious schism between what is officially anointed as standard practice and what is in fact the standard practice in later discussion of the invention paradigm for HCI.

1.2 Lack of theory

The guiding hope in doing evaluation work is that the data collected and the methods developed can cumulate into coherent analyses about *why* some systems and techniques are more usable than others, and about *how* to enhance the usability of future systems and techniques. It's a bottom-up approach to developing theory. However, directly contrasting two complex situations (e.g., two versions of a system) to determine which one is better is a poor vehicle for sorting out and saving experience. Complex alternatives with no a priori theoretical analysis do not become interpretable merely in virtue of a simple horse race. It would take an infinity of such "one-off" contrasts to build a theory from the bottom up. Even the simple and controlled situations studied in experimental psychology would be intractably indeterminate without top-down theoretical direction.

Many of the difficulties with direct contrast evaluations can be attributed to this lack of theory. The use of toy-scale problem domains and simple, quantitative measures is problematic in that without a theory of HCI domains there is no way to know whether a toy problem is representative of a real problem or not. There is no way to know whether one is studying a coherent part of the real problem, or an accidental and idiosyncratic case. Can an analysis of writing 50-line programs be scaled up to the problem of writing 5,000-line programs? Is the task of pointing a cursor at an arbitrary screen location a coherent part of the task of pointing a cursor in the course of editing text? Are interpretations of isolated system events related to interpretations of the very same events embedded in a real stream of user interaction? Answering such questions is impossible without a theory with which to interpret the toy situations and to extrapolate from them to real situations.

Sheil (1981), for example, noted that complexity is not linear with program length. It certainly seems that the task of editing a 5,000-line program raises problems of navigation and naming conventions that are just not raised in the task of editing a 50-line program. Elements of HCI situations may interact and trade off in different ways as the problem scale or the task changes. Is avoiding GOTO statements more or less important that employing indentation in a program listing? And are there contexts in which the relation is inverted? Again, without a theory there is no way to extrapolate these interactions. Indeed, one can do little more than organize separate studies on the basis of superficial features (e.g., as pertaining to variable names or menu systems). Without a theory of, for example, how people understand, name, and remember entities, there is no way to work back from a variety of performance differences

obtained in a variety of experimental settings to an explanation of the underlying concepts that caused the differences (see Newell, 1973).

In the absence of a theoretical framework for understanding usability, HCI evaluation work has had to address issues at a very large grain of analysis. Hauptmann and Green (1983), for example, contrasted a natural language interface with a menu interface for creating business graphics (failing to find any significant differences in time, errors or attitudes). Of course, contrasting natural language with menus is painting with a rather broad stroke: how could a single experimental contrast resolve such a multifaceted contrast? Were the two interfaces individually optimized to be the best interface possible in their respective interface styles? Were they controlled to have the same functional capabilities and the same task-relative functional capabilities? The same kinds of questions arise for the examples discussed earlier, evaluating structured programming, object oriented programming and symbolic notations. The lack of theory forces these crude contrasts; but the crude contrasts prohibit pertinent or univocal results.

Methods and theories in software technology are often collections of loosely connected prescriptions. Ideas like structured programming and direct manipulation (Shneiderman, 1983) are important theoretical concepts, and they surely carry empirical consequences. But they are not falsifiable in the Popperian sense (Popper, 1965): one cannot hope to reject such ideas *tout court* on the basis of isolated laboratory tests, to try to do so is to get the logic of the inquiry wrong. From our current perspective of a few years hence, it is clear that no outcome of the Sheppard et al. (1979) study could have rejected structured programming as an appropriate prescriptive theory. The real evaluation need is for detailed qualitative information that can guide the revision and integration of such ideas. The issue is not whether structured programming is good, or indeed whether it is better than some other approach; the issue is what structured programming really consists in, how in detail it impacts actual programming tasks, and how it can be integrated into routine programming practice.

The assessment goal is just too limiting: a paradigm that merely evaluates distinctions articulated by others, deprives itself of playing any directive role (Sheil, 1981). In this context, we can understand why studies like Sheppard et al. failed to lead to the development of an articulated theory of programming: the evaluation enterprise bound itself to what already existed, commenting at a high level on the appropriateness of specific techniques from the mid-1970s. A poignant example is the work showing that input error rates are reduced when using teletype terminals instead of visual display units (Walther and O'Neil, 1974; Carlisle, 1970). It was never a possibility that teletype terminals would supplant visual display units through the course of technological evolution, quite the contrary. The bald evaluation result, without specific implications for the design of future visual display devices, can only be seen as an historical curiosity.

Empirical evaluation of software and systems is a key to usability. But it is a separate question whether a *science* of human-computer interaction can arise out of this activity. In fact, it did not. The evaluation paradigm introduced psychology and psychologists to the HCI problem domain. It was a platform for establishing the importance of usability and for developing empirical approaches to measuring the usability of systems and software. However, its methodological commitments and lack of theory cast it in a supporting role in emerging software and user interface science: more of a commentator on new technology than a directive force. The challenge that this raised was how psychology could play a more directive role in the development of new software and user interface technology.

2. Cognitive Description

In the early 1980s there was a shift toward bringing HCI research under the aegis of broader psychological theory. Shneiderman (1980: 51), for example, used Miller's (1956) classic paper on human information processing limitations to derive the prescription that programmers avoid the use of GOTO constructs. Shneiderman analyzed the process of understanding programs as involving the recoding of lines of code into meaningful "chunks". GOTO jumps in a program text disrupt this structure by functionally chunking nonadjacent lines of code. In 1983, Card, Moran and Newell (1983) published a compelling monograph adapting information processing

psychology to the description of fluent user interaction with text editors. These efforts had an enormous effect, enlarging and intensifying interest in the psychology of usability both within computer science *and* within psychology.

This shift confronted one of the key limitations of earlier work, the lack of theory. Tying specific empirical results to theories of human information processing provided means to integrate diverse results, to resolve nonsignificant or conflicting findings, to dampen the distortions of poor research, but most importantly to develop abstractions that, in principle, could help lead the development of software technology and user interface design.

However, this work also raised new issues and problems. Aligning HCI phenomena with cognitive descriptions of those phenomena is useful to the extent that the cognitive descriptions themselves are rich, revealing and well-integrated. In fact, psychological theory is at least as fragmented as software theory and methodology. Building a psychology of usability by placing this body of fragmented theory into correspondence with software situations risks inheriting the fissures as well as the solid ground. Ironically, cognitive description work also threatened the major achievement of human factors evaluation, namely, establishing the centrality of direct usability testing to the ultimate success of computing systems and techniques. The cognitive description paradigm entrained a strongly analytic conception of software design, raising the question of how much direct evaluation might be necessary if a good theory were in hand.

2.1 Breadth versus depth

Scientific psychology seeks to understand behavior and experience by providing laws, concepts, and explanations. However, there are severe limits on what types of phenomena psychology can address with these goals and tools; there are ranges over which the goals and tools make sense and outside of which they do not. In particular, academic psychology typically attempts to capture generalizations across domains. But fine details of specific task situations can be very important: what a person thinks and decides to do is often ascribable to knowledge of a single fact, e.g., the name of a particular command in a particular system. These fine-grained details serve as boundary markers for theorizing: scientific laws that must refer to individual facts as conditions seem unwieldy, and psychologists routinely make a strategic retreat to abstract or artificial domains to control such details.

This is a reasonable heuristic, with extensive precedent in the sciences. Classical mass point mechanics is developed under the idealization of frictionless contact, even though there are no frictionless systems. Other theoretical apparatus has been developed to add back the effects of friction in real systems. The difficult details of friction are treated as "perturbations" of the classical theory (Gleick, 1987). Similarly, the traditional research strategy in psychology has been to focus on sweepingly general issues and distinctions under the idealization that domain and situation context can be ignored. Basic psychological research addresses topics like the "structure of memory," but not, for example, "memory for Unix commands" (Norman, 1981). It tries to generally resolve "big" issues like "is there a separate mental type for imagery?" (Pylyshyn, 1973; Paivio, 1971).

It turns out that describing frictionless contact provides a useful foundation for understanding the motion of real objects in real circumstances. Even though the effects of friction are not simple, treating these effects as perturbations of an idealized theory has also proven tractable in engineering applications (for example, computing trajectories). The question is whether the same basic strategy is useful in psychology. This is an open question. Newell (1973), for example, criticized the pursuit of sweeping dichotomies like existence of a separate mental type for imagery, saying "you can't play twenty questions with nature and win." Indeed, the emergence in the 1980s of Cognitive Science as a broader discipline, incorporating psychology with the serious consideration of the structure of task domains, can be seen as a response to traditional idealizations (Carroll, 1988a).

Chase and Simon's (1973) classic study of expertise in chess showed that, for a reconstructive memory task, chess masters tended to recall piece positions in attack and defense groupings. This study has had two very different legacies. On the one hand, it opened up a variety of questions about domains. How are chess piece groupings indexed in a player's

memory; how they are accessed in realistic tasks (like playing chess, as opposed to reconstructive memory for arbitrary board positions), how does expertise in chess develop through significant spans of time? Many of these issues have been pursued and in a variety of domains (see Chi, Glaser and Farr, 1988), though many would argue that the work still takes too narrow a view of the process of attaining expertise and of the nature of expert knowledge and performance (e.g., Dreyfus and Dreyfus, 1986).

On the other hand, Chase and Simon's result was sweepingly generalized as "experts have chunks," and has been mechanically replicated in domain after domain. There is no rich and well-integrated theory of either experts or chunks outside of considerations of specific domains. Thus, these studies show only that when humans know something about a domain and are asked to do reconstructive memory tasks of an arbitrary sort, they use what they know to do the task. A series of these studies have been undertaken in HCI contrasting memory performance for scrambled and unscrambled program listings (Adelson, 1981; McKeithen, Reitman, Reuter, and Hirtle, 1981, Shneiderman, 1980). This work showed that people with programming experience can use knowledge of language structures in organizing their memories.

This finding has not led to rich understandings of how people achieve expertise in programming or about how programming knowledge is indexed in memory and accessed in performance. It has not helped to guide the development of new software tools and environments. These cognitive descriptions do not address and provide no guidance in practical aspects of programming (the design of programming languages, environments, education, etc.); they do not even engage issues specific to the domain of programming (the types of modules one would want in a library to facilitate code reusability).

An extensive tradition of psychological research describes learning, memory and error patterns for paired-associates, the classic nonsense syllable (e.g., Esper, 1925; Postman and Stark, 1962). This work has been applied to the analysis of user performance with various types of command languages (Barnard, Hammond, Morton, Long and Clark, 1981; Carroll, 1982; Landauer, Galotti and Hartwell, 1983). For the most part, these applications have been no less mechanical than those of the "experts have chunks" work. Yet they have been relatively more successful in that the cognitive descriptions developed for command language interactions have had fairly specific prescriptive content for command language design. Indeed, HCI research on command names has led to specific revisions in philosophical and linguistic conceptions about what names are (Carroll, 1985).

But this work, and indeed all cognitive description work in HCI, is subject to a very fundamental problem in the underlying logic of the inquiry. Psychology concerns itself with *existence*: is there a separate mental type for imagery? HCI, like any applied science domain, concerns itself with *impact*: how much of a difference will certain types of consistency make in the learnability of a command language? This is why the "experts have chunks" work seems reasonable from the perspective of our curiosity about chess masters and other experts, but difficult to apply in the face of questions about how to support experts and facilitate the development of expertise. This is also why the use of extreme contrasts, like scrambled programs versus structured programs, can make sense in the pursuit of basic theory, but much less so in the pursuit of meaningful application.

Landauer (1987a) has recently called attention to this in observing that while basic psychology routinely focusses on the "significance" of effects, it typically disregards the *size* of effects. Cognitive descriptions framed in terms of existence dichotomies can be assessed by the statistical significance of direct contrasts: do expert programmers chunk more than novices? However, such differences do not guarantee that the effects will be large enough to matter. Would it matter if experts reliably chunked 2 percent more than novices? Would it matter if scrupulously consistent command languages were learned 3 percent faster than randomly consistent languages? To determine the practical size of effects one needs to consider cost-benefit tradeoffs in realistic tasks. Chunking may have a big effect on people trying to memorize scrambled little programs, but the size of effect question forces attention to real programmers writing and reading real programs. The two situations might be quite different.

2.2 *Design by deduction*

HCI is fundamentally a design domain: it exists in the first place because of the need to design more usable computing artifacts for people to use. Design in a complex and poorly charted domain can seem like trial and error. How should user interface design work proceed to ensure more usable user interfaces? The human factors evaluation paradigm sought to address this kind of question by providing methodology for directly evaluating design techniques (like structured programming) and particular artifacts (for example, a particular programming language or programming environment). But direct evaluation operates on a case by case basis. The cognitive description paradigm sought to improve upon this by providing theoretical abstractions beyond the specific cases (see Moran, 1981).

Card, Moran and Newell (1983) made what is surely the most thorough and disciplined attempt to interpret and develop modern information processing psychology into a foundation for the design of computer systems. In their GOMS model (an acronym for Goals, Operators, Methods and Selection rules), users hierarchically decompose their goals into successively finer subgoals until these match a basic set of methods. The user has rules for selecting methods appropriate to the current situation, and each method itself consists of a sequence of operators, keypresses and hand motions. This analysis was fitted to a variety of text editing performance data, in many cases yielding consistent values for the model's parameters.

However, the theory proved quite limited in application to user interface design. GOMS was not able to describe problem-solving activity, only routine, over-practiced performance. In fact, it could not describe errors at all, even though nearly a third of the routine behavior it sought to describe consisted of error and error recovery. It was also severely hampered by the race between function and usability: By the time it had produced good performance descriptions for error-free, over-practiced behavior on line-oriented editors, the focus of concern in user interfaces and end-user applications had moved on to other problem areas. (See Carroll and Campbell, 1986, for further discussion.) The work had its greatest impact on relatively low-level aspects of human-computer interaction, like the analysis of pointing devices (Card, English and Burr, 1978). Indeed, it appears that this approach may only work for user interaction events on the order of one second in duration in which errors are extremely rare and/or extremely regular (!), and for technological contexts that are unchanging on the order of decades (Newell and Card, 1985). Few design problems in HCI fall into this rather severe category.

Most cognitive description work is far less theoretically ambitious than the GOMS work. For example, the use of menu selection, as an alternative to typed commands is sometimes "deduced" from the fact that humans are better at recognition than at recall (e.g., Tennant, Ross and Thompson, 1983). This is terribly oversimplified. Users of menu systems must deal with formidable navigation problems (MacGregor and Lee, 1987; Robertson, McCracken, and Newell, 1981). They must deal with complex morphological, semantic and referential relations *between* various selection names (Carroll, 1985). Here again, the evolution of user interface technology is complicating the simple dichotomies: rich aliasing (Gomez and Lochbaum, 1985) may substantially mitigate the relative difficulty of recall and alternative approaches to menu design may carry differing performance implications (pop-up menus, multiple selection menus, active forms). Finally, though the advantage of recognition over recall is an established sweeping principle in psychology (e.g., Crowder, 1976), Black and Sebrechts (1981) have observed that there are circumstances in which the reverse is true.

We earlier considered Shneiderman's (1980) reference to Miller's (1956) analysis of human information processing limitations in grounding the prescription to avoid GOTOs. Miller's specific argument, however, does not consider spatial or temporal proximity of items to be "chunked." Accordingly, the GOTO prescription cannot be deduced from Miller's analysis. Indeed, virtually nothing of much interest could be *deduced* from the specifics of Miller's analysis. The connection is more informal: Miller's work called attention to the (obvious) fact that humans are limited with respect to the information they can manage; Shneiderman was inspired by this to suggest a particular tactic for easing information management in programming. The informality of the theoretical linkages is not specially problematic: the non-psychological theory-components of HCI do no better (e.g., what is an interface toolkit?). Having theories

cogent enough and pertinent enough to even informally direct and inspire design work is a big advantage.

The problem vis-a-vis design by deduction is that in none of these examples of cognitive description applied to design do we have in hand the ancillary theoretical apparatus to deductively bridge between the "leading claims" and the implementation details. GOMS *is* probably a reasonable first approximation framework for thinking about task analysis. Recognition probably *is* easier than recall in many circumstances. GOTOs probably *do* strain human information processing capacity. But to use this theoretical material deductively in design we need to know precisely how the details of given situations interact with and modulate the psychological principles. None of the theories is complete enough to tell us this. Hence none can be used deductively.

To an extent, this lack can be addressed through theory development. For example, Polson (1987) has developed the GOMS approach into a potentially more useful design tool. However, other considerations indicate that HCI design can never be rendered deductive. The particular complexity of software technology stems from the fact that everything inherently interacts with everything else (Brooks, 1987). The technological context plays an important role in determining whether an idea will survive at all. For example, object oriented techniques have been seen as a major advance in software technology, but the successful use of these techniques is limited by the availability of appropriately supportive programming environments (Uebbing, 1987). Many times these interactions cannot be anticipated at all. Presenting rich information displays and direct access to running code often entrains cluttered displays and inefficient performance. Many of these critical details and interactions cannot be analyzed before a prototype system is built. Indeed, one of the most important determinants of the success of software technologies is their amenability to revision and reimplementation on hardware and software platforms not even available when they were first developed (Brooks, 1987).

The cognitive description paradigm in HCI was a genuine advance. It provided independent conceptual foundations for the psychology of HCI that made it possible to develop useful theory. Reciprocally, it brought the HCI domain within the purview of academic psychologists. This has opened a two-way dialog within which basic cognitive psychology may stand to gain as much from the cognitive engineering case study of HCI as HCI may stand to gain from the science of cognition (Carroll, 1987b; Norman, 1987).

3. Usability-Innervated Invention

The human factors evaluation and cognitive description paradigms share basic assumptions about the position of psychological analysis in HCI. They assume that psychology operates *outside* the development process, outside even the research prototyping process. They assume that the role of psychologists in HCI is to offer *commentary*: evaluations, theoretical descriptions, but not direct participation in the invention, design and development of new HCI technologies and artifacts. This assumed positioning and role for psychology in HCI is all the more striking when one recognizes that HCI is fundamentally a design domain. HCI is *about* designing new software tools and user interfaces. Seen in this light, the traditional paradigms for psychology in HCI have pursued a tangential, supporting role in the field's key endeavor and *raison d'etre*.

It has, of course, been recognized that serious usability research needs to pay serious attention to the nature of HCI domains and tasks. This concern has always been in the focus of HCI work. But being relevant to designer needs is not the same as taking the initiative in the design work itself. The implicit division of labor in HCI has had chronic organizational consequences. For example, a recent panel discussion at the ACM CHI'88 Conference asked how human factors specialists, and cognitive scientists working on usability, can organize to effectively work with designers and developers (Grudin, 1988). The answers offered are revealing: human factors professionals should be placed directly into development groups, human factors professionals should *manage* the developers, usability consultants from outside the organization should be used (!). The traditional paradigms created an organizationally adversarial basis for the exchange of commentary between software developers and psychologists.

The traditionally assumed positioning and role of psychology within HCI is now being seriously questioned. In this new paradigm of "usability-innervated invention," usability is seen as connecting the invention of HCI artifacts to user needs no less essentially than nerves connect organs and muscle tissues to sensory and motor brain centers. The activity of muscles and organs is meaningful only insofar as it is innervated by sensation and action; the activity of inventing HCI artifacts is meaningful only insofar as it is innervated by usability considerations. Conversely, sensory and motor centers exist at all to innervate the body's muscle and organs; understanding usability is important because it produces the critical direction for HCI invention. In this view, HCI artifacts are not merely evaluated or described in terms of their usability; *they are conceived and created for usability.*

3.1 Psychology as a mother of invention

Building and inventing things is not a traditional activity in psychological research. Psychology is part natural science and part social science; its traditional focus is the analysis of natural and social phenomena. In the technological arena of HCI, this traditional focus was straightforwardly extended to the analysis of technology through evaluation and theoretical description. But these traditional activities also provided the opportunity for psychologists working in HCI domains to develop technological skills and domain experience. In many cases, these psychologists are now in a position not only to *analyze* usability problems, but to *synthesize* technological solutions. In his plenary address at the CHI + GI'87 Conference, Tom Landauer (1987b) succinctly captured this in casting "psychology as a mother of invention" in HCI.

Many recent prototype systems and interface techniques were invented by psychologists to instantiate specific psychological claims and to allow these claims to be explored and developed empirically. For example, Landauer's group analyzed human performance in a variety of naming and reference tasks to develop specific tools and techniques for keyword information systems (e.g., Furnas, Landauer, Gomez and Dumais, 1983). The database system Rabbit (Williams, 1984) and its "retrieval by elaboration" paradigm embodied claims about the structure of human memory and memory search as consisting in the manipulation of concrete exemplars. The variety of "Minimalist" training materials and software environments described in Carroll (1988b) embody a set of claims about how new users learn computer applications. The display management system Rooms (Card and Henderson, 1987) embodies an analysis of typical user working sets (services and data accessed simultaneously).

User interface metaphors are a systematic and detailed intrusion of psychology into modern computing system development (Carroll and Thomas, 1982; Carroll, Mack and Kellogg, 1988). For example, systems that provide electronic workspaces that can be written to and viewed by multiple users in a cooperative interaction session are presented as "chalkboard" systems in the way that they are described to users and even in the way they appear and operate (Stefik, Foster, Bobrow, Kahn, Lanning and Suchman, 1987). Thinking of the system as a physical chalkboard provides an initial familiarity for the user. It also suggests specific tasks and approaches to accomplishing them. It provides the user with an initial conceptual vocabulary within which to couch questions and draw conclusions. (Analogous points could be made for other new computer interface designs ranging from task oriented window layout (Carroll, Herder and Sawtelle, 1987), to object oriented programming (Rosson and Alpert, 1988)).

Many recent structure-directed editors and intelligent tutoring systems for programming are clearly vehicles for instantiating psychological analyses of programming tasks and learning. For example, analyses of programming plans (e.g., Soloway and Ehrlich, 1984) are embodied in the Bridge tutor (Bonar and Liffick, 1987). Analyses of how students learn to program in Lisp (Anderson, Farrell and and Sauers, 1984) have been embodied in a variety of intelligent tutoring systems for teaching Lisp (Anderson and Skwarecki, 1986; Reiser et al., 1988). Indeed, Anderson has recently (1987) argued that designing and evaluating computer tutors provides unique advantages to *basic,* academic psychological research into the mental procedures and knowledge that comprise human cognition.

Of course, psychologists *per se* are not always the inventors, but psychological rationale routinely plays a determining role in the invention of new software technology. In this work,

HCI transcends merely serving as an arena for *applying* empirical experience and theoretical analysis to invention. A better description is that a two-way relationship has developed in which HCI artifacts themselves are treated as media for codifying experience and analysis, in which HCI theories are "applied invention" no less than HCI artifacts are "applied theory" (Carroll and Campbell, 1988). For example, the theoretical development of the concept "direct manipulation" (Shneiderman, 1983) devolved from a collection of specific HCI inventions. But this constitutes a radical shift in the underlying ontology of HCI, namely, seeing computer artifacts like interface metaphors, menu hierarchies, programming paradigms and languages, tutors, and the like as playing theory-like roles.

One standard role of theories is to codify empirically falsifiable claims (Popper, 1965). Artifacts embody testable claims about how users can understand and make use of system function in a medium that makes appropriate empirical investigations possible. Each command name, each icon, each menu makes claims about the ways users think about the tasks they will undertake with these systems.

These claims are mutually interrelated, creating a sort of web of theory more intricate and more comprehensive than any analysis deducible from conventional discursive psychological theory. A piece of software, like the Unix operating system, makes a huge number of specific claims about what command names, operations, and so forth will be convenient for users. These claims can be *wrong* (see Norman, 1984). Desktop interfaces make myriad claims about familiar presentation and natural conceptual vocabularies, about clipboards, stationery pads, folders, waste baskets -- about how these objects behave and interact. Moreover, the leading claims, for example as integrated within a metaphor like the desktop, have myriad specific dependencies on a diverse set of ancillary claims (for example, claims inherent in the presentation of highlighting, preferences, and scrolling elevators).

Empirical theories provide explanations by placing logical and causal constraints on phenomena. Artifacts support explanations of the form "this specific feature has this specific usability consequence." The "Tear Off" command in the early Lisa desktop system provides an example. In this system, "Tear Off" spawns a new instance from a prototype object: Tear Off stationery applied to a stationery pad creates a piece of stationery. The command was a menu selection, not a gesture (Move is an example of a gestural command: one selects with the pointer and then moves by moving the pointer). Thus, there was a sort of inconsistency between Move and Tear Off. Some users initially tried to Tear Off by selecting and then rapidly sweeping the pointer (making a tearing gesture). This error has little consequence, and proved relatively easy for users to sort out on their own. A more difficult problem stemmed from the fact that Tear Off also applied to non-pad objects like folders: the user needed to Tear Off from a "folder pad" to get a new folder (Carroll and Mazur, 1986).

Theories also contribute to the development of science by providing useful foundations for further theorizing. Artifacts facilitate theoretical development in the sense that given artifacts make task analyses possible that in turn facilitate the invention and development of new artifacts. The typewriter metaphor was a critical step in the development of the desktop metaphor, which in turn has been critical in the development of newer interface metaphors such as rooms and task maps. Understanding user problems at this level of qualitative detail can be of immediate use in the design of new software artifacts. Indeed, in subsequent desktop interface products the Tear Off command evolved into a Make New Folder command.

Theories enable and compel greater explicitness in empirical claims. This is part of the traditional motivation to formalize. Artifacts serve this role in a manner quite analogous to classical views of simulation (Fodor, 1968; Newell and Simon, 1972). To paraphrase Newell and Simon, both must "perform" the claims they incorporate: the implementation details must be made explicit, which can lead to further learning about the nature of the claims being made. Simulations, however, are used by psychologists, for specific research purposes; artifacts are used by a wide range of people to do real work. Simulations are interpreted and evaluated by criteria of *descriptive adequacy* (Chomsky, 1965): a simulation of problem-solving behavior may be judged on the basis of how closely it fits the sequence of moves in a verbal protocol, whether it predicts all and only the kinds of errors that are observed, etc. Artifacts are interpreted and evaluated by criteria of *usability*.

Simulations are usually seen as convenient vehicles for theories, but not as *necessary*. Are artifacts merely convenient expressions of HCI theories, or do they play a more fundamental role? This question cannot be answered now, but it seems likely that artifacts are in principle irreducible to a more conventional theory medium. The reason for this, if it is so, would be the unbounded interrelation of the many claims inherent in a computer artifact, the fact that everything in software seems to impact everything else (Brooks, 1987), the fact that details of context and situation critically impinge upon the usability of systems (Whiteside and Wixon, 1987; Winograd and Flores, 1986; Suchman, 1987). All these may be views of the same underlying state of affairs: the design of software may be of an order of complexity beyond that which conventional theories can explain or predict (Hayek, 1967).

In the introduction, we considered the apparent paradox that product innovations in user interface design often *lead* HCI research rather than following from it in the conventionally assumed flow of "technology transfer" from Research to Development. However, the view of HCI in which its artifacts play theory-like roles in organizing research defuses the perplexity of this state of affairs. Empirical research often follows the explicit codification of theories. In HCI the medium of choice for expressing theories of usability is in many cases an exemplary artifact. The appearance of such an artifact predictably stimulates empirical research.

3.2 Ecological analysis

The paradigm of usability-innervated invention has many consequences for the traditional empirical roles of psychologists working in HCI domains. There are consequences both for what kinds of situations are studied and for what kinds of information are sought in empirical studies. In both areas, the driving considerations devolve from invention. The model of research practice in experimental psychology, originally adapted to HCI through human factors evaluation, has been augmented by the requirement that empirical work bear more directly on the invention and development of new artifacts. In this sense, current work is shifting toward greater responsiveness to the ecology of HCI as an ecology of invention, design and development.

Ecologically responsive empirical analysis of HCI domains takes place *in vivo*: in software shops, more often than in psychological laboratories. It addresses *whole* problems, *whole* situations, when they are still technologically current, when their resolution can still constructively impact the direction of technological evolution. Its principal goal is the discovery of design requirements, not the verification of hypothesized direct empirical contrasts or cognitive descriptions. A recent example is Curtis, Krasner and Iscoe's (1988) study of the software design process. The detailed interviewing of real designers produced specific technical proposals for improving software tools and the coordination of project management, an assessment of major bottlenecks, and a new framework for thinking about software design as a learning and communication process. (See Nielsen, Mack, Bergendorff and Grischkowsky, 1986, and Rosson, Maass and Kellogg, 1988, for similar kinds of studies.)

Carroll and Campbell (1988) characterized HCI invention in terms of the "task-artifact cycle": a given understanding of the tasks programmers need to and want to accomplish helps to define objectives for new software artifacts (languages, environments and education, etc.) to support them in these tasks. Any artifact fundamentally alters the tasks for which it was designed, raising the need for further task analysis, and in time for the design of further artifacts, and so on. An example is the progression from user interfaces based on the typewriter metaphor to those based on the desktop. Early word processing applications were designed to exploit specific knowledge their users already had about typewriting, function keys, data display, command names and so forth (Carroll and Thomas, 1982).

The typewriter metaphor, however, altered office tasks and in doing so helped to open up technological possibilities by preparing users for further electronic office applications (calculators, calendars, mail, database). This evolution in office task expectations and understandings was better addressed by systems employing the desktop metaphor. However, desktop systems also presented a variety of specific problems and possibilities to users (Carroll and Mazur, 1986; Whiteside et al., 1985). This further task analysis has again helped to define further interface

artifacts, new metaphors for display organization in user interfaces ("rooms," Card and Henderson, 1987; "task paths," Carroll, Herder and Sawtelle, 1987).

To constructively operate within the task-artifact cycle, HCI empirical work must provide rich analyses of real users working on real tasks. The main research setting for such ecological analysis is the case study. A case study can begin and end anywhere in the task-artifact cycle; the key requirement is access to real situations. Case study task analysis usually consists of the collection of detailed, qualitative information (thinking aloud protocols, interviews). Such data are arbitrarily rich: they can be returned to over and over again, and analyzed from many different perspectives. A typical approach is to make videotapes to create a vivid and permanent data library. The development of Minimalist training materials and software environments, cited earlier, was based on such case study analysis (Carroll, 1988b). Mack's (1987) inventory of new user expectations about cause and effect relationships in the operation of a word processor was a case study analysis culminating in the development of a prototype that more intuitively presented word processing function.

It is important to collect information over a significant span of time to eliminate ephemeral effects. Monitoring patterns of actual use of a software environment often supplements the more direct interview and protocol techniques. Wixon, Whiteside, Good and Jones (1983) analyzed patterns of spontaneous interaction with an electronic mail application to determine how to design a more usable command interface for the application. Kelley (1984) analyzed the desk calendars of office workers to determine requirements for an electronic calendar facility. Gould and Boies and their collaborators have designed a series of voice messaging systems using this approach (Gould and Boies, 1983; Gould, Boies, Levy, Richards and Schoonard, 1987).

The key goal of ecological task analysis in the task-artifact cycle is to produce requirements for subsequent design work. This places emphasis on identifying big factors -- big needs, big usability problems. Thus, one typical output of this phase is an error taxonomy, a qualitative description of what is giving the user trouble, how it is happening, what users are doing in consequence, etc. The complexity and rapid evolution of software technology requires richer and more open-ended methods than the direct contrast testing of the human factors evaluation and cognitive description approaches. This richer style of task analysis is interpretive, inductive; it seeks to discover, not merely to confirm or disconfirm.

It often requires studying user interface technologies and applications *before* they are even developed: after all, that's the point at which empirical guidance can be most effectively directive (Carroll and Campbell, 1986). For obvious reasons, it is difficult to do such work, but a variety of simulation techniques have been developed. For example, Gould, Conti and Hovanyecz (1983) simulated a speech recognition capability to explore technological tradeoffs in a technology that was not then available. Carroll and Aaronson (1988) analyzed interactions with a simulated intelligent help facility to help direct the development of more usable artificial intelligence applications.

To help direct the task-artifact cycle, new types of usability data and new roles for usability data are being developed. For example, since the ideas that lead HCI research typically become codified in products first, it is important to be able to interpret running systems, to extract key ideas and work with them. Norman (1984) made an influential psychological interpretation of key aspects of the Unix operating system. Carroll and Mazur (1986) analyzed new user expectations and experiences using the on-line tutorial and direct manipulation interface of the Lisa system. Rosson and Alpert (1988) have recently analyzed psychological implications of objected oriented design. Carroll, Mack and Kellogg (1988) outlined tools for analyzing user interface metaphors in design.

Another focus for the development of tools for empirical analysis is the process of software and system development. A comprehensive methodology of goal definition and measurement has been developed for guiding the discovery of appropriate usability requirements and evaluating progress toward meeting these requirements within the design process (Bennett, 1984; Carroll and Rosson, 1985; Whiteside, Bennett and Holtzblatt, 1988).

Usability-innervated invention offers a more directive role in framing new applications and user interfaces, and a more ecologically responsive role for empirical work. It incorporates and builds upon the prior orientations of human factors evaluation and cognitive description, but

pushes onward in taking more seriously the fact that HCI is a design field, that it exists to invent more usable systems and software. Earlier approaches to psychology in HCI had in effect isolated the task analysis part of the task-artifact cycle from the definition, development and first use of new software and user interface technology, because of preconceptions about the kinds of contributions psychologists might make to HCI. As a result, and in addition to a variety of specific limitations discussed above, they offered only commentary on the process and products of design, not participation.

4. The Ecology of Computing

The progression of three paradigms in the recent history of HCI comprises a case study of a field discovering what it is about. HCI has achieved much by exploiting the context of its own practice. It has assimilated the evaluation methodology of experimental psychology, the theory of cognitive science, and the invention and development of new technology. Each step in this evolution has solved some of problems posed by the step preceding it.

The emerging paradigm of usability-innervated invention redresses the ecological limitations of direct contrast laboratory evaluations by promoting new methods and new roles for empirical evaluation. It redresses the theoretical limitations of design by deduction by countenancing richer sources and embodiments of scientific theory. This in turn has resolved other puzzles about HCI. For example, the primacy of product development ideas in HCI research is puzzling only until it is recognized that product development is a major context for HCI research: one of the important roles of psychology in HCI is to provide interpretation and conceptual clarification for product innovations.

Even the mysterious race between function and usability dissolves: appropriately contextualized HCI research cannot lag the technological leading edge; it lives at the technological leading edge; indeed, it creates the technological leading edge. For example, there is no race between usability and function in the development of the Rooms display management system (Card and Henderson, 1987), even though the Rooms approach is at the edge of our current understanding of display management tasks and artifacts. The race between function and usability is simply an untoward side-effect of the organizational consequences of human factors evaluation and cognitive description.

Usability-innervated invention offers a new basis for these organizational dynamics. When the basis for collaboration is evaluative or descriptive commentary offered from outside the design team, the grounds are frequently political, and power-based, or *interpreted* as political and power-based. This is completely unconstructive: it pushes empirical evaluation and psychological theory further away from invention. Operating within the task-artifact cycle as task analysts, as inventors of artifacts, offers a deeper source of interdisciplinary and inter-organizational coordination: shared understanding of what the problems are, why the current design situation is what it is, what the immediate and longer-term options are, and how they trade off. It offers the alternative of committed, cooperative work.

4.1 Science and invention

There is a conventional view of the relationship between scientific research and the invention, design and development of practical artifacts. The idea is that basic science provides an understanding of nature which can then be applied deductively in practical contexts. The relationship between science and invention in HCI, as it has emerged through the course of the last 15 years, is interesting from this standpoint in that appears to be culminating (at least to this point in time) somewhat unconventionally.

To be sure, the conventional view was what the field started out with: the vision of the human factors evaluation and cognitive description paradigms was to develop an empirical basis, to develop a theoretical framework and finally to apply the theory deductively in design. Through hard experience, HCI discovered that things were not this neat. Invention produces theory in HCI at least as much as it applies theory and this has fundamentally altered the nature of the empirical work. The resolution of this may lie in a countercurrent in the history of science, questioning the conventional view itself. For example, Hindle (1981) analyzed a variety of 19th

century inventions and failed to find any deductive grounding in the basic science of the time. Hindle suggests that the conventional view may have developed as recently as the 1850s in the American scientific establishment as a tactic for increasing the prestige of and federal support for basic research.

Many well-known instances of invention clearly do not conform to the conventional view. The pulley, for example, had been used effectively for some 2,000 years before an adequate scientific analysis of its operation was developed within Newtonian mechanics. The violins of the 17th century were so finely crafted that their design was merely emulated for over 200 years. Indeed, only in the last couple of decades has there been any appreciable acoustic understanding of how violins really work (Hutchins, 1962). And it is not clear yet whether the science of acoustics itself was more a contributor to or a beneficiary of this work.

Of course, there *is* a relation between basic science and invention, but not a simple deductive relation. Gomory (1983) puts the point well when he argues that the development of technology is both more complex and less predictable than the basic research from which it is seen to spring. Gomory discusses the first 150 years of technology development for the steam engine. He shows that the "revolutionary" engines of the mid-nineteenth century actually evolved through many small steps, each relying on the chance availability of a technological niche, an application in which the technology could survive and develop. The case study of HCI suggests that the relation between basic science and invention can be highly interactive and reciprocal. The conventional view goes wrong in trying to frame this relation too narrowly.

It is a commonplace of the philosophy of science since positivism to observe that there are no "discovery procedures," no algorithms to carry us from the raw material of empirical science to a theoretical explanation of that raw material. A way to put this point is to say analogously that there are no "invention procedures": the logical leap from basic data and theory to the invention and development of a usable artifact is neither more or less deterministic than the step we are more familiar with, namely the step from the raw material of experience to a theory of a conventional sort. The applied science of the conventional view is a myth.

Psychology is a young science, so is Computer Science, so is Cognitive Science, and above all, so is HCI. But this raises the question of whether the complex and reciprocal interaction of science and invention in HCI is attributable just to the youth of the relevant fields, to scientific growing pains as it were. In view of this possibility it is relevant to consider the acoustic analysis of the violin as conducted over the past 40 years by members of the Catgut Society, an interdisciplinary group of musicians, instrument craftsmen, physicists and engineers. Carla Maley Hutchins, the senior member of this team, told me an interesting anecdote about an early stage in her collaboration with Bell Labs physicists. The physicists' initial approach was to disassemble a violin, induce sine waves and measure resulting resonances.

It's a beautiful image; it recalls the direct contrasts of human factors evaluation and the shallow theories of cognitive description. It recalls models of error-free user behavior as bases for understanding how to design usable computer systems and applications. It is the conventional strategy of divide and conquer, which too often requires subtracting out the essence of the problem being solved. Inducing pure sine waves into the pieces of the violin to measure the resonances is not an adequate approach to understanding the violin. The sound to which a real violin responds is not a pure sine wave and it is not induced; it is a complex tone produced by bowing. Moreover, the resonances in a whole violin derive both from the parts and from the composition of the parts, indeed from the big chunk of air trapped within the composition of the parts. Analyzing the parts, does not add up to an understanding of the behavior of the whole.

The point is not that these idealized acoustic analyses were pointless. Such work is on-going, and has even produced techniques useful in violin-making (Hutchins, 1981). And the point is not that acoustic science has nothing to offer as a foundation for understanding violins (bowing does not produce pure sine waves, but it does produce sound after all). The point is that even in physics the initial approach to applying science to design is often simplified and inadequate, whereas the effective role is more interactive and reciprocal. Indeed, the comparison can pushed much further: the research of the Catgut Society eventuated in the design and development of a new set of stringed instruments, the Violin Octet. The analysis could go only so far when its purview was an account of the standard string quartet (which acoustically is a very

accidental collection of instruments). To develop and assess laws of acoustic scaling, to test and develop claims about the violin, it was necessary to build novel instruments (Hutchins, 1967; Hutchins and Schelleng, 1967).

The violin is intrinsically a very appealing example. But one needn't go so far. Anyone in the New York area recalls the renovation of Carnegie Hall. There was much concern and much debate about the impact this would have on the famous acoustics of that hall. Acoustics, the old science of physics, could not deductively direct or predict the outcome. Indeed, to this day the only fact that everyone agrees on is that the acoustics of Carnegie Hall are now different.

4.2 The current perplexity

Failure to appreciate the subtleties of technology development, coupled with the inherent limitations of the human factors evaluation and cognitive description paradigms of HCI and the emergence of the usability-innervated invention paradigm, has caused substantial perplexity in the field. One body of work has responded to Newell and Card's (1985) worry that psychology must be scientifically hard to survive in HCI by retreating into the study of low-level phenomena and of highly constrained situations, creating a very insular research microcosm. One of the key areas of its focus is replicating classic phenomena from the psychology of nonsense list learning (e.g., Polson, Kieras and Muncher, 1987). This approach flaunts all the limitations of the cognitive description paradigm. It is not at all clear that it can be relevant to HCI design work.

Another body of work has rejected psychology as a totally inappropriate foundation for design work in HCI (Whiteside and Wixon, 1987; Winograd and Flores, 1986). In this view, focussing on models of the mind and conceiving of people as computational devices that process inputs, generate goal lists, and then execute plans and responses all merely obscure and obstruct the designer's most important responsibility and objective: to understand the user's needs and wishes and to serve the user. This work flaunts the theoretical limitations of human factors evaluation, looking to hermeneutics as a conceptual foundation for design and emphasizing interpretations that are unique to the situation and to the individual doing the interpreting, and explicitly discouraging model-building or any form of abstraction. However, since it is bound to particular cases, this work cannot provide any framework for understanding HCI phenomena as types.

Both approaches are dismal in prospect: one offering no hope of practical impact and the other no hope of understanding. However, from the standpoint of the present discussion these extreme positions have despaired too quickly. An orderly evolution of HCI work has produced a paradigm that builds upon the genuine contributions of human factors evaluation and cognitive description and at the same time redresses their limitations with respect to design impact and the ecological validity of empirical work.

HCI has often been described as an "interdisciplinary" research area, but only now are the full interdisciplinary possibilities emerging. Participating fully and in a variety of roles in the evolution of computer technology offers psychologists in HCI a uniquely creative opportunity. It's a demanding opportunity. Inventing the future is more difficult than commenting on it. Pushing psychological theory to interpret and analyze new technological situations and embodying psychological claims and results in HCI artifacts is not easier than evaluating finished systems, computing t-tests and calculating performance times. But then one does not move to the frontier for the comforts of familiarity. The possibility and the challenge of HCI today is to move forward to new roles and new ideas in technology and science.

Note

This paper is derived from lectures given at Teachers College, Columbia University, the University of Michigan, the University of Western Ontario, and the IBM Watson Research Center in the winter of 1988, and from collaborative discussions with Robert Campbell and Elliot Soloway. I am grateful to John Black, Norman Brown, John Karat, Wendy Kellogg, John Gould, Joan Roemer, Mary Beth Rosson, Linda Tetzlaff and Zenon Pylyshyn for comments on the talks and on earlier versions of this chapter.

References

Adelson, B. (1981). Problem solving and the development of abstract categories in programming languages. *Memory and Cognition, 9,* 422-433.

Anderson, J.R. (1987). Methodologies for studying human knowledge. *Brain and Behavioral Sciences, 10(3),* 467-505. (With commentary).

Anderson, J.R., Farrell, R. and Sauers, R. (1984). Learning to program in Lisp. *Cognitive Science, 8,* 87-129.

Anderson, J.R. and Skwarecki, E. (1986). The automated tutoring of introductory computer programming. *Communications of the ACM, 29,* 842-849.

Barnard, P.J., Hammond, N.V., Morton, J., Long, J.B. and Clark, I.A. (1981). Consistency and compatibility in human-computer dialog. *International Journal of Man-Machine Studies, 15,* 87-134.

Bennett, J.L. (1984). Managing to meet usability requirements: Establishing and meeting software development goals. In J. Bennett, J. Sandelin and M. Smith (Eds.), *Visual display terminals.* Englewood Cliffs, NJ: Prentice-Hall, pages 161-184.

Bonar, J.G. and Liffick, B.W. (1987). A visual programming language for novices. University of Pittsburgh Technical Report LSP-5.

Black, J.B. and Sebrechts, M.M. (1981). Facilitating human-computer communication. *Applied Psycholinguistics, 2,* 149-177.

Brooks, F.P. (1987). No silver bullet: Essence and accidents of software engineering. *IEEE Computer, 20/4,* 10-19.

Card, S.K., English, W.K., and Burr, B.J. (1978). Evaluation of mouse, rate-controlled isometric joystick, step keys, and task keys for text selection on a CRT. *Ergonomics, 21/8,* 601-613.

Card, S.K. and Henderson, D.A. (1987). A multiple virtual-workspace interface to support user task switching. In J.M. Carroll and P.P. Tanner (Eds.), *Proceedings of CHI+GI'87: Human Factors in Computing Systems and Graphics Interface.* (Toronto, April 5-9). New York: ACM, pages 53-59.

Card, S.K., Moran, T.P. and Newell, A. (1983). *The psychology of human-computer interaction.* Hillsdale, NJ: Erlbaum.

Carlisle, J.H. (1970). Comparing behavior at various computer display consoles in time-shared legal information. Rand Corporation, Report No. AD712695. Santa Monica, CA.

Carroll, J.M. (1982). Learning, using and designing command paradigms. *Human Learning: Journal of Practical Research and Applications, 1,* 31-63.

Carroll, J.M. (1985). *What's in a name? An essay in the psychology of reference.* New York: W.H. Freeman.

Carroll, J.M. (1987a). Five gambits for the Advisory Interface Dilemma. In M. Frese, U. Ulich, and W. Dzida (Eds.) *Psychological issues of human computer interaction in the work place.* Amsterdam: North Holland, pages 257-274.

Carroll, J.M. (Ed.) (1987b). *Interfacing thought: Cognitive aspects of human computer interaction.* Cambridge, MA: Bradford Books/M.I.T. Press.

Carroll, J.M. (1988a). Modularity and naturalness in cognitive science. *Metaphor and Symbolic Activity, 3(2),* 61-86.

Carroll, J.M. (1988b). *The Nurnberg funnel: Designing minimalist instruction for practical computer skill.* Englewood Cliffs, NJ: Prentice Hall.

Carroll, J.M. and Aaronson, A.P. (1988). Learning by doing with simulated intelligent help. *Communications of the ACM, xx*, pages.

Carroll, J.M. and Campbell, R.L. (1986). Softening up Hard Science: Reply to Newell and Card. *Human-Computer Interaction, 2*, 227-249.

Carroll, J.M. and Campbell, R.L. (1988). Artifacts as psychological theories: The case of human-computer interaction. IBM Research Report RC 13454. Yorktown Heights, NY.

Carroll, J.M., Herder, R.E. and Sawtelle, D.S. (1987). TaskMapper. In H.J. Bullinger and B. Shackel (Eds.) *Human-Computer Interaction: Proceedings of INTERACT'87* Amsterdam: North Holland, pages 973-978.

Carroll, J.M., Mack, R.L. and Kellogg, W.A. (1988). Interface metaphors and user interface design. In M. Helander (Ed.) *Handbook of Human-Computer Interaction*. Amsterdam: North Holland.

Carroll, J.M. and Mazur, S.A. (1986). LisaLearning. *IEEE Computer, 19/11*, 35-49.

Carroll, J.M. and Rosson, M.B. (1985). Usability specification as a tool in interactive development. In H. Hartson (Ed.) *Advances in Human-Computer Interaction 1*, Norwood, NJ: Ablex, pages 1-28.

Carroll, J.M. and Soloway, E. (1988). The evolving role of software psychology in software development practice.

Carroll, J.M. and Thomas, J.C. (1982). Metaphor and the cognitive representation of computing systems. *IEEE Transactions on Systems, Man and Cybernetics, 12*, 107-115.

Chapanis, A. 1959. *Research techniques in human engineering*. Baltimore, MD: The Johns Hopkins Press.

Chase, W.C. and Simon, H.A. (1973). Perception in chess. *Cognitive Psychology, 4*, 55-81.

Chi, M.T., Glaser, R. and Farr, M.J. (Eds.) (1988). *The nature of expertise*. Hillsdale, NJ: Erlbaum.

Chomsky, A.N. (1965). *Aspects of the theory of syntax*. Cambridge, MA: MIT Press.

Crowder, R.G. (1976). *Principles of learning and memory*. Hillsdale, NJ: Erlbaum.

Curtis, B. (1980). Measurement and experimentation in software engineering. *Proceedings of the IEEE, 68/9*, 1144-1157.

Curtis, B. (Ed.) (1985). *Human factors in software development*. Washington, D.C.: IEEE Computer Society Press.

Curtis, B. (1986). By the way, did anyone study any real programmers? In E. Soloway and S. Iyengar (Eds.) *Empirical studies of programmers*. Norwood, NJ: Ablex, pages 256-262.

Curtis, B., Krasner, H. and Iscoe, N. (1988). A field study of the software design process for large systems.

Dijkstra, E.W. (1972). Notes on structured programming. In O.J. Dahl, E.W. Dijkstra and C.A.R. Hoare (Eds.) *Structured programming*. New York: Academic Press, pages 1-82.

Dreyfus, H.L. and Dreyfus, S.E. (1986). *Mind over machine: The power of human intuition and expertise in the era of the computer*. New York: The Free Press.

Esper, E.A. (1925). A technique for the experimental investigation of associative interference in artificial linguistic material. *Language Monographs, 1*, 1-47.

Fodor, J.A. (1968). *Psychological explanation*. New York: Random House.

Furnas, G.W., Landauer, T.K., Gomez, L.M., and Dumais, S.T. (1983). Statistical semantics: Analysis of the potential performance of key-word information systems. *The Bell System Technical Journal*, 1753-1806.

Gleick, J. 1987. *Chaos: Making a new science*. New York: Viking.

Gomez, L.M. and Lochbaum, C.C. (1985). People can retrieve more objects with enriched key-word vocabularies. But is there a performance cost? In B. Shackle (Ed.) *Human-Computer Interaction -- INTERACT'84*. Amsterdam: North Holland, pages 257-261.

Gomory, R.E. (1973). Technology development. *Science, 220*, 576-580.

Gould, J.D. and Boies, S.J. (1983). Human factors challenges in creating a principal support office system -- The Speech Filing System approach. *ACM Transaction on Office Information Systems, 1(4)*, 273-298.

Gould, J.D., Boies, S.J., Levy, S., Richards, J.T., and Schoonard, J. (1987). The 1984 Olympic Message System: A case study of system design. *Communications of the ACM, 30,* 758-769.

Gould, J.D., Conti, J., and Hovanyecz, T. (1983). Composing letters with a simulated listening typewriter. *Communications of the ACM, 26/4,* 295-308.

Gould, J.D. and Lewis, C.H. (1985). Designing for usability: Key principles and what designers think. *Communications of the ACM, 28(3),* 300-311.

Green, P. (1987). Tips on writing a good paper proposal. *Computer Systems Technical Group Bulletin, 14,* 6-10.

Grudin, J. 1988. Integrating human factors in software development. In E. Soloway, D. Frye and S. Sheppard (Eds.) *CHI'88 Conference on Human Factors in Computing Systems* New York: ACM (May 15-18, Washington, D.C.), pages 157-160.

Hauptmann, A.G. and Green, B.F. (1983). A comparison of command, menu-selection and natural language computer programs. *Behaviour and Information Technology, 2,* 163-178.

Hayek, F.A. (1967). The theory of complex phenomena. In F.A. Hayek (Ed.) *Studies in philosophy, politics, and economics.* Chicago: University of Chicago Press.

Hindle, B. (1981). *Emulation and invention.* New York: New York University Press.

Holt, R.W., Boehm-Davis, D.A. and Schultz, A.C. (1987). Mental representations of programs for student and professional programmers. In G.M. Olson, S. Sheppard and E. Soloway (Eds.), *Empirical studies of programming: Second workshop.* Norwood, NJ: Ablex, pages 33-46.

Hutchins, C.M. (1962). The physics of violins. *Scientific American,* November (Reprint 289).

Hutchins, C.M. (1967). Founding a family of fiddles. *Physics Today, 20(2),* February.

Hutchins, C.M. (1981). The acoustics of violin plates. *Scientific American, 245(4),* October, 170-186.

Hutchins, C.M. and Schellung, J.C. (1967). A new concert violin. *Journal of the Audio Engineering Society, 15(4).*

Kelley, J. F. (1984). An iterative design methodology for user-friendly natural language office information applications. *ACM Transactions on Office Information Systems, 2,* 26-41.

Landauer, T.K. (1987a). Relations between cognitive psychology and computer system design. In J.M. Carroll (Ed.) *Interfacing thought: Cognitive aspects of human-computer interaction.* Cambridge, MA: Bradford/MIT Press, pages 1-25.

Landauer, T.K. (1987b). Psychology as a mother of invention. In J.M. Carroll and P.P. Tanner (Eds.), *Proceedings of CHI+GI'87: Human Factors in Computing Systems and Graphics Interface.* (Toronto, April 5-9). New York: ACM, pages 333-335.

Landauer, T.K., Galotti, K.M. and Hartwell, S. (1983). Natural command names and initial learning: A study of text editing terms. *Communications of the ACM, 26,* 495-503.

Ledgard, H., Whiteside, J.A., Singers, A., and Seymour, W. (1980). The natural language of interactive systems. *Communications of the ACM, 23,* 556-563.

Liebelt, L.S., McDonald, J.E., Stone, J.D. and Karat, J. (1982). The effect of organization on learning menu access. *Proceedings of the Human Factors Society, 26th Annual Meeting,* pages 546-550.

Love, T. (1977). *Relating individual differences in computer programming performance to human information processing abilities.* Ph.D. Dissertation, University of Washington.

Mack, R.L. (1987). Understanding and learning text-editing skills: Observations on the role of new user expectations. In S. Robertson, J. Black and W. Zachary (Eds.), *Cognition, Computing and Cooperation.*

McKeithen, K.B., Reitman, J.S., Reuter, H.H., and Hirtle, S.C. (1981). Knowledge organization and skill differences in computer programmers. *Cognitive Psychology, 13,* 307-325.

Moran, T.P. (1981). The command language grammar: A representation for the user interface of interactive computer systems. *International Journal of Man-Machine Studies, 15,* 3-50.

Miller, G.A. (1956). The magical number seven plus or minus two: Some limits on our capacity for processing information. *Psychological Review, 63,* 81-97.

Murrel, S. (1983). Computer communication system design affects group decision making. In A. Janda (Ed.), *Proceedings of CHI'83 Human Factors in Computing Systems.* (Boston, December 12-15). New York: ACM, pages 63-67.

Newell, A. (1973). You can't play twenty questions with nature and win. In W. Chase (Ed.) *Visual information processing*. New York: Academic Press.

Newell, A. and Card, S.K. (1985). The prospects for psychological science in human-computer interaction. *Human-Computer Interaction, 1*, 209-242.

Newell, A. and Simon, H.A. (1972). *Human information processing*. Englewood Cliffs, NJ: Prentice-Hall.

Nielsen, J., Mack, R.L., Bergendorff, K., and Grischkowsky, N.L. (1986). Integrated software usage in the professional work environment: evidence from questionnaires and interviews. In *Proceedings of CHI'86 Human Factors in Computing Systems* (Boston, April 13-17), ACM, New York, 162-167.

Norman, D.A. (1981). The trouble with Unix. *Datamation, 27*, 556-563.

Norman, D.A. (1987). Cognitive Engineering -- Cognitive Science. In J.M. Carroll (Ed.) *Interfacing thought: Cognitive aspects of human-computer interaction*. Cambridge, MA: Bradford/MIT Press, pages 323-336.

Paivio, A. (1971). *Imagery and verbal processes*. New York: Holt, Rinehart & Winston.

Polson, P. (1987). A quantitative theory of human-computer interaction. In J.M. Carroll (Ed.) *Interfacing thought: Cognitive aspects of human-computer interaction*. Cambridge, MA: Bradford/MIT Press, pages 184-235.

Polson, P., Kieras, D., and Muncher, E. (1987). Transfer of skills between inconsistent editors. Microelectronics and Computer Technology Corporation Technical Report ACA-HI-395-87, Austin, Texas.

Popper, K. (1965). *Conjectures and refutations*. New York: Harper and Row.

Postman, L. and Stark, K. (1962). Retroactive inhibition as a function of set during the interpolated task, *Journal of Verbal Learning and Verbal Behavior, 10*, 44-51.

Pylyshyn, Z. (1973). What the mind's eye tells the mind's brain: A critique of mental imagery. *Psychological Bulletin, 80*, 1-24.

Reiser, B.J., Friedman, P., Gevins, J., Kimberg, D.Y., Ranney, M. and Romero, A. (1988). A graphical programming language interface for an intelligent Lisp tutor. Princeton University CSL Report 15.

Robertson, G., McCracken, D., and Newell, A. (1981). The ZOG approach to man-machine communication. *International Journal of Man-Machine Communication, 14*, 461-488.

Rosson, M.B. and Alpert, S. (1988). Cognitive implications of object oriented programming.

Rosson, M.B., Maass, S., and Kellogg, W.A. (1988). The designer as user: Building requirements for design tools from design practice.

Sheil, B.A. (1981). The psychological study of programming. *ACM Computing Surveys, 13*, 101-120.

Sheppard, S.B., Curtis, B., Millman, P., and Love, T. (1979). Modern coding practices and programmer performance. *IEEE Computer, 12/12*, 41-49.

Shneiderman, B. (1980). *Software psychology: Human factors in computer and information systems*. Cambridge, MA: Winthrop.

Shneiderman, B. (1983). Direct manipulation: A step beyond programming languages. *IEEE Computer, 16(8)*, 57-69.

Shneiderman, B. and McKay, D. (1976). Experimental evaluations of computer program debugging and modification. *Proceedings of the 6th International Congress of the International Ergonomics Association*, July.

Soloway, E. and Ehrlich, K. (1984). Empirical studies of programming knowledge. *IEEE Transactions on Software Engineering, SE-10(5)*, 595-609.

Stefik, M., Foster, G., Bobrow, D., Kahn, K., Lanning, S. and Suchman, L. (1987). Beyond the chalkboard: Computer support for collaboration and problem solving in meetings. *Communication of the ACM, 30*, 32-47.

Suchman, L. (1987). *Plans and situated actions*. Cambridge: Cambridge University Press.

Tennant, H.R., Ross, K.M. and Thompson, C.W. (1983). Usable natural language interfaces through menu-based natural language understanding. In A. Janda (Ed.), *Proceedings of CHI'83 Human Factors in Computing Systems*. (Boston, December 12-15). New York: ACM, pages 154-160.

Uebbing, J. Panel on making products. In L. Power and Z. Weiss (Eds.), Proceedings OOPSLA'87: Object-Oriented Programming Systems, Languages and Applications. Special issue of Sigplan Notices, vol. 23, no. 5, 1988 page 105-111 (Orlando, FL, October 4-8, 1987).

Walther, G.H. and O'Neil, H.F. (1974). On-line user-computer interface: the effects of interface flexibility, terminal type, and experience on performance. *Proceedings of the National Computer Conference, 43,* Montvale, NJ: AFIPS Press.

Weissman, L. (1974). *A methodology for studying the psychological complexity of computer programs.* Ph.D. Dissertation, University of Toronto.

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.

Whiteside, J., Jones, S., Levy, P.S. and Wixon, D. (1985). User performance with command, menu, and iconic interfaces. In L. Borman and B. Curtis (Eds.) *Proceedings of CHI'85: Human Factors in Computing Systems.* (San Francisco, April 14-18) New York: ACM, pages 185-191.

Whiteside, J. and Wixon, D. (1987). Improving human-computer interaction -- a quest for cognitive science. In J.M. Carroll (Ed.) *Interfacing thought: Cognitive aspects of human-computer interaction.* Cambridge, MA: Bradford/MIT Press, pages 337-352.

Williams, M.D. (1984). What makes RABBIT run? *International Journal of Man-Machine Studies, 16,* 405-438.

Winograd, T. and Flores, F. (1986). *Understanding computers and cognition: A new foundation for design.* Norwood, NJ: Ablex.

Wixon, D., Whiteside, J., Good, M., and Jones, S. (1983). Building a user-defined interface. In A. Janda (Ed.), *Proceedings of CHI'83 Human Factors in Computing Systems.* (Boston, December 12-15). New York: ACM, pages 24-27.