Annu. Rev. Psychol. 1997. 48:61–83

HUMAN-COMPUTER INTERACTION: Psychology as a Science of Design

John M. Carroll

Computer Science Department and Center for Human-Computer Interaction, Virginia Tech, Blacksburg, Virginia 24061-0106

KEY WORDS: usability, design, user models, applied psychology, human factors

ABSTRACT

Human-computer interaction (HCI) study is the region of intersection between psychology and the social sciences, on the one hand, and computer science and technology, on the other. HCI researchers analyze and design specific user interface technologies (e.g. pointing devices). They study and improve the processes of technology development (e.g. task analysis, design rationale). They develop and evaluate new applications of technology (e.g. word processors, digital libraries). Throughout the past two decades, HCI has progressively integrated its scientific concerns with the engineering goal of improving the *usability* of computer systems and applications, which has resulted in a body of technical knowledge and methodology. HCI continues to provide a challenging test domain for applying and developing psychological and social theory in the context of technology development and use.

CONTENTS

0084-6570/97/0201-0061$08.00

THE EMERGENCE OF USABILITY

Human-computer interaction (HCI) has emerged relatively recently as a highly successful area of computer science research and development and of applied psychology. Some of the reasons for this success are clearly technical. HCI has evoked many difficult problems and elegant solutions in the recent history of computing, e.g. in work on direct manipulation interfaces, user interface management systems, task-oriented help and instruction, and computer-supported collaborative work. Other reasons for its success are broadly cultural: The province of HCI is the view the nonspecialist public has of computer and information technology and its impact on their lives; HCI is the visible part of computer science. The most recent reasons for the success of HCI are commercial: As the underlying technologies of computing become commodities, inscribed on generic chips, the noncommodity value of computer products resides in applications and user interfaces, that is, in HCI.

HCI has evolved rapidly in the past two decades as it has struggled to develop a scientific basis and utility in system and software development. In this chapter, I review the progression of HCI toward a science of design. My touchstone is Simon's (1969) provocative book *The Sciences of the Artificial*. The book entirely predates HCI, and many of its specific characterizations and claims about design are no longer authoritative (see Ehn 1988). Nevertheless, two of Simon's themes echo through the history of HCI and still provide guidance in charting its continuing development.

Early in the book, Simon discussed the apparently complex path of an ant traversing a beach, observing that the structure of the ant's behavior derives chiefly from the beach; the ant pursues a relatively simple goal and accommodates to whatever the beach presents. The external world, including technology human beings create, should be expected to play a powerful role in structuring human behavior and experience. Late in the book, Simon sounds the second theme: the need for a science of design, a research paradigm and university curriculum directed at understanding, furthering, and disseminating design knowledge. He lamented the tendency of engineering disciplines to adopt goals and methodologies from the natural sciences, to their detriment with respect to design.

HCI is a science of design. It seeks to understand and support human beings interacting with and through technology. Much of the structure of this interaction derives from the technology, and many of the interventions must be made through the design of technology. HCI is not merely applied psychology. It has guided and developed the basic science as much as it has taken direction from it. It illustrates possibilities of psychology as a design science.

Software Psychology

The work that constitutes the historical foundation of HCI was called "software psychology" in the 1970s (e.g. Shneiderman 1980). The goal then was to establish the utility of a behavioral approach to understanding software design, programming, and the use of interactive systems, and to motivate and guide system developers to consider the characteristics of human beings. Software psychology had two distinct methodological axioms. The first assumed the validity of a received view of system and software development, namely the so-called waterfall model of top-down decomposition and discretely sequenced stages with well-specified hand-offs (e.g. Royce 1970). The second assumed two central roles for psychology within this context: (*a*) to produce general descriptions of human beings interacting with systems and software, which could be synthesized as guidelines for developers; and (*b*) to directly verify the usability of systems and software as (or more typically, after) they were developed.

Software psychology inaugurated a variety of technical projects pertaining to what we now call the *usability* of systems and software: assessing the relative complexity of syntactic constructions in programming languages (e.g. Sime et al 1973), classifying people's errors in specifying queries and procedures (Miller 1974), describing the utility of mnemonic variable names and in-line program comments (Weissman 1974), and explicating how flowcharts serve as a programming aid (Shneiderman et al 1977). This work inspired many industrial human-factors groups to expand the scope of their responsibilities to include support for programming groups and the usability of software.

The basic axioms of software psychology proved to be problematic. The waterfall idealization of design work is infeasible and ineffective (e.g. Brooks 1975/1995). It is only observed when enforced, and it is best regarded as a crude management tool for very large-scale, long-term projects. As computer research and development diversified in the 1970s and 1980s, small and distributed personal work organizations became more commonplace. Product development cycles were often compressed to less than a year.

The two roles assigned to software psychologists were also problematic and resulted in a division of labor. Researchers, mainly in universities, developed general descriptions of users and framed them as general guidelines. Human-factors specialists in industry tried to apply these guidelines in specific projects. This division did not work particularly well. From the standpoint of practical goals, the research of this period tended to focus on unrepresentative situations (e.g. undergraduates standing in for programmers, 50-line programs standing in for business systems, and teletypes standing in for display tubes). To obtain statistically stable results, researchers often created outrageous con-

trasts (organized versus disorganized menus, structured versus scrambled programs). The researchers sometimes understood little about the users of their guidelines and proffered superfluous advice.

Psychologists and others playing the human-factors specialist role in industry were frustrated trying to use and encourage use of these guidelines. They were also frustrated in their other role of verifying the usability of finished systems, because the research tools they had (formal experiments aimed at specific differences among alternatives) were too costly and too uninformative to allow them to serve as anything more than gatekeepers. Human-factors specialists were often seen as imposing bureaucratic obstacles blocking heroic developers.

The origins of HCI in software psychology posed two central problems for the field in the 1980s. One problem was to better describe design and development work and to understand how it could be supported. The other problem was to better specify the role that psychology, in particular, and social and behavioral science, more broadly, should play in HCI.

Iterative Development

Starting in the 1970s, empirical studies of the design process began to explicate the difficulties of the waterfall model. Design work is frequently piecemeal, concrete, and iterative. Designers may work on a single requirement at a time, embody it in a scenario of user interaction to understand it, reason about and develop a partial solution to address it, and then test the partial solution—all quite tentatively—before moving on to consider other requirements. During this process, they sometimes radically reformulate the fundamental goals and constraints of the problem. Rather than a chaos, it is a highly involuted, highly structured process of problem discovery and clarification in the context of unbounded complexity (e.g. Carroll et al 1979, Curtis et al 1988, Malhotra et al 1980).

The leading idea is that designers often need to do design to adequately understand design problems. One prominent empirical case was Brooks's (1975/1995) analysis of the development of the IBM 360 Operating System, one of the largest and most scrupulously planned software design projects of its era. Brooks, the project manager, concluded that system and software designers should always "plan to throw one away" (pp. 116–23). This was a striking lesson to draw and carried with it many implications. For example, formal and comprehensive planning and specification aids (such as detailed flowcharts) will have limited utility in supporting such an iterative design process.

This reformulation of design created an opening for new ideas. Noteworthy inspiration came from the work of the great industrial designer Henry Dreyfuss, who had pioneered an empirical approach in the 1940s (Dreyfuss 1955).

Dreyfuss's approach institutionalizes an accommodation to designers' propensity for concrete, incremental reasoning and testing. It incorporates four central ideas: (*a*) early prototyping with (*b*) the involvement of real users, (*c*) introduction of new functions through familiar "survival forms," and (*d*) many cycles of design iteration. Dreyfuss pushed beyond the designer's need for prototyping and iteration as a means of clarifying the design problem (also emphasized by Brooks 1975/1995) to the user's knowledge, experience, and involvement to constrain design solutions.

A typical example is Dreyfuss's design work on airplane interiors for Lockheed. Dreyfuss sent two associates back and forth across the United States on commercial flights to inventory passenger experiences. They found that passengers were often baffled by design details like water taps that were unnecessarily novel. They were impressed that people wanted to think of airplane seats as armchairs and not as "devices." Initial designs were prototyped in a Manhattan warehouse, and a full flight of "passengers" was hired to occupy the mock-up for 10 hours: to store carry-on luggage, to eat meals, to use lavatories. These tests concretized requirements for seating, storage space, lavatory-door latches, and so forth, and permitted low-cost, iterative reworking of the original designs.

In the 1980s, the inevitability of an empirical orientation toward system and software design rapidly evolved from a somewhat revolutionary perspective to the establishment view. This development provided early and critical motivation and direction for research on user interface management systems to enable prototyping (Tanner & Buxton 1985), it encouraged user participation in design (Gould & Boies 1983, Nygaard 1979, Pava 1983), it emphasized user interface metaphors for presenting novel functionality through familiar concepts (Carroll & Thomas 1982, Smith et al 1982), and it made "rapid prototyping" a standard system development methodology (Wasserman & Shewmake 1982, 1985).

Iterative development shifted the focus of usability evaluation from summative to formative (Scriven 1967). Formal experiments are fine for determining which of two designs are better on a set of a priori dimensions, but they are neither flexible nor rich enough to guide continual redesign. "Thinking aloud" had been pioneered by deGroot (1965) and Newell & Simon (1972) in the study of puzzles and games, and it had been shown to vividly illuminate strategic thought (Ericsson & Simon 1985). In the 1980s, thinking aloud became the central empirical, formative evaluation method in HCI (e.g. Mack et al 1983, Wright & Converse 1992).

User Models

The second problem area bequeathed to HCI by software psychology was to characterize a robust science base that could underwrite system development.

The cornerstone in this effort was the GOMS project of Card et al (1983). GOMS (Goals, Operators, Methods, and Selection rules) provided a framework for systematically analyzing the goals, methods, and actions that comprise routine human-computer interactions. This was an advance on prior human-factors modeling, which did not address the cognitive structures underlying manifest behavior. Indeed, it was an advance on the cognitive psychology of the time: It explicitly integrated many components of skilled performance to produce predictions about real tasks. At first, GOMS appeared to promise a comprehensive paradigm for scientifically grounded HCI design (Newell & Card 1985, but cf Carroll & Campbell 1986). The actual impact of these models has been more narrow, although they have been usefully applied in domains where user performance efficiency is the critical usability variable [e.g. modeling telephone operator scripts (Gray et al 1992)].

One salient limitation of GOMS models is that they do not describe learning, yet the problems of new users beginning with computer systems was perhaps *the* technical focus of the 1980s. The learning problem was conceived as a matter of coordinating knowledge in two domains: the task and the device. The user learns mappings between goals in the task domain and actions in the device domain, between events in the device domain and effects in the task domain (Moran 1983, Payne et al 1990). Much of this discussion focused on the "consistency" of mappings from commands and other user interface actions to application functions (e.g. Payne & Green 1989). For example, learning a command named "pull" is facilitated by a complementary command named "push" (versus, say, one named "grab"); commands like "edit data_file" and "remove data_file" are mutually facilitative (see also Esper 1925). Consistency turned out to be highly intentional (Carroll 1985), significantly idiosyncratic (Furnas et al 1983), and even questionable as a general design objective (Grudin 1989). Nevertheless, promoting consistency in user interface and application design remains a prominent practical issue (Nielsen 1989), and these models are the foundation for our understanding of it.

The role of prior knowledge in learning to use computer systems was another focus of user modeling work. It was widely noted that new users tried to understand computers as analogical extensions of familiar activities and objects (e.g. Douglas & Moran 1983, Mack et al 1983). This observation led to a variety of user interface "metaphors," such as the now pervasive desktop interface, and a paradigm for user interface control and display called "direct manipulation," which was gradually articulated through the 1980s (Hutchins et al 1986, Shneiderman 1983). It also led to theories of user interface metaphor (Carroll et al 1988).

A second limitation of the early GOMS-style cognitive models is that they do not address nonroutine problem solving and error. Studies showed that new

users spent a third to a half of their time in error recovery (e.g. Mack et al 1983). More significantly, these studies showed that often the dispositions that make people good sense makers are also causes for characteristic learner problems (Carroll 1990): People want to learn by doing, but this inclines them to opportunistically jump around in sometimes brittle learning sequences. They want to reason things out and construct their own understandings, but they are not always planful, and they often draw incorrect inferences. They try to engage and extend prior knowledge and skill, which can lead to interference or overgeneralization. They try to learn through error diagnosis and recovery, but errors can be subtle, can tangle, and can become intractable obstacles to comprehension and to motivation.

This work entrained the conception of the "active user," improvising, hypothesizing, trying to make sense of a very complex environment. It led to emphases on designing for learning-by-doing and for error (Carroll 1990, Lewis & Norman 1986). For example, Carroll & Carrithers (1984) created a "training wheels" interface in which attempted departures from a correct action path are blocked. Users are informed that the action is not appropriate and are permitted to try again without penalty. This kind of design supports sense-making but not necessarily efficient performance. The active user was somewhat of an alternative to the GOMS conception of the user as an information processor.

USER-CENTERED SYSTEM DEVELOPMENT

In the early days of HCI, the notion that computer systems and software should be designed and developed with explicit consideration of the needs, abilities, and preferences of their ultimate users was not taken seriously. Most writings about computing from the mid-1970s are stunningly dismissive of usability and rather patronizing of users. After only a decade, the computer industry and the discipline of computer science were transformed. The case had been made for a user-centered system development process, a process in which usability was a primary goal. People began to distinguish sharply between technology-driven exploratory development, which is now often accompanied by explicit disclaimers about usability, and "real" system development, in which empirically verified usability is the final arbiter.

With the advent of the 1990s, HCI research had become relatively well integrated in computer science. A 1988 Association for Computing Machinery (ACM) task force listed HCI as one of nine core areas of the computer science discipline (Denning et al 1989). A joint curriculum task force of the ACM and the Institute of Electrical and Electronic Engineers (IEEE) recommended the inclusion of HCI as a common requirement in computer science programs (Tucker & Turner 1991). HCI was included as one of ten major sections of the

first *Handbook of Computer Science and Engineering* (Tucker 1997). In the 1990s, computer science students and the corporations that hire them are demanding HCI courses in university curricula. Several major computer science departments have designated HCI as a research focus. Two comprehensive undergraduate texts have appeared (Dix et al 1993, Preece et al 1994).

In industry, HCI practitioners have become well integrated in system development. HCI specialists have moved into a great variety of roles beyond human-factors assurance. They have been routinely included in customer/user interactions to understand the need for new products, product planning, and specification; the development and evaluation of prototypes; the design of documentation and training; and installation and user support. There has been an obvious trend for HCI specialists to be promoted into project management. None of these trends impugns the psychological nature of HCI; rather, they indicate that it has been a practical success. In addition, they remind us that successful applied work involves more than merely applying lab-based theories and results, a theme that continues to be articulated in the current era.

HCI remains an emerging area in computer science. As an applied area of social and behavioral science, it continues to broaden. The issues raised in the early days of software psychology are still being resolved and elaborated: How can iterative development be supported and improved? How should we manage resources in iterative development to optimize cost benefit? How should we expand the scope and richness of the early cognitive user models? How can we cumulate and develop technical lessons learned in iterative development? What role can HCI play in broadening, grounding, and invigorating the development of the social and behavioral science base?

These issues have been pursued through refinements in the original notions of iterative development and user models, discussed below in the sections Usability Engineering, Design Rationale, and Cooperative Activity.

Usability Engineering

Iterative development is consistent with the real nature of design. It emphasizes the discovery of new goals, the role of prototyping and evaluation, and the importance of involving diverse stakeholders, including users. But what makes iterative development more than merely well-intentioned trial and error?

"Usability engineering" became the banner under which diverse methodological endeavors were carried out in the 1980s (Nielsen 1993, Whiteside et al 1988). There were three key notions: First, it was proposed that iterative development be managed according to explicit and measurable objectives, called "usability specifications" (Carroll & Rosson 1985; see also Bennett 1984, Butler 1985). Thus, in designing a word-processing program, one would iteratively design and redesign, prototype and evaluate, include real secretaries

in the design deliberations, but also make explicit commitments to precisely operationalized goals such as "two thirds of users will be able to prepare a two-page business letter in less than ten minutes with fewer than three errors after 30 minutes training." Usability specifications are now standard practice in HCI development.

The second key notion in usability engineering was a call to broaden the empirical scope of design. A variety of approaches and techniques for user participation were developed, many of which emphasized "low-tech," cooperative activities to facilitate collaboration between users, who bring expertise on the work situation, and developers, who bring expertise on technology (Greenbaum & Kyng 1991, Kuhn & Muller 1993, Schuler & Namioka 1993). This approach went beyond prior formulations of user involvement by describing broader and more active roles for users. In "participatory design," as the approach is now known, users are involved in setting design goals and planning prototypes, instead of becoming involved only after initial prototypes exist.

Field-study approaches to characterizing the users' real needs and circumstances also became prominent (Whiteside & Wixon 1987). This emphasis came to be known as "contextual design" (Wixon et al 1990). It overturned the laboratory bias inherited by HCI from cognitive psychology by arguing that laboratory situations are often not controlled simulacrums of real situations but are better regarded as distinct but eccentric situations in their own right. Note that field-study approaches are not equivalent to user participation: Often field studies bring to light facts in the background of the context of use, circumstances of which the users themselves are unaware. Conversely, field studies cannot reveal the perspectives and insights users bring to the development process as designers.

In the early 1990s, contextual design converged with a line of ethnographic research that had produced edifying descriptions of use contexts, but which had generally eschewed direct involvement in design (e.g. Suchman 1987). This combination, sometimes called "ethnographically-informed design" (Bentley et al 1992), has become quite prominent. Like participatory design, it pushes a specific methodological commitment a step further by advocating very detailed observation of behavior in real situations. In practice, ethnographically informed design is frequently aimed at characterizing unarticulated power relations, organizational assumptions, and practical know-how that organize the workplace (Blomberg 1995). A complementary stream of work, sometimes called "conversation analysis," addresses itself to finer-grained behavioral structure (Greatbatch et al 1995). The commitment to revelatory interpretation of situated behavior sometimes engenders a behaviorism that gives too little weight to what people experience and report (Nardi 1995b).

The third key notion in usability engineering was cost effectiveness. It is expensive to carry out many cycles of prototyping, evaluation, and redesign. Developers need to employ efficient methods throughout and to know when they have reached diminishing returns. In the 1980s, much HCI work was directed at creating better prototyping tools and environments. An early theme in this work was the *separation* of user interface software from application software (i.e. system functionality) to modularize redesign of the user interface in user interface management systems (e.g. Tanner & Buxton 1985). Subsequent work emphasized the *coordination* of user interface and application software development to facilitate the creation of task-transparent user interfaces (Sukaviriya et al 1993, Szekely et al 1993). Concerns about ease of *implementation* motivated a family of prototyping tools based on the premise that user interface software could be directly created "by demonstration" (Cypher et al 1993).

The issue of cost effectiveness also guided methodological work on usability evaluation. Indeed, the earliest refinement of the GOMS model was a keystroke-counting approximation (Card et al 1983). Frequently, methodological shortcuts were grounded in the exigencies of system development. For example, thinking aloud protocols had become a standard empirical method, but they were generally not analyzed at the level of detail typical in cognitive psychology. Often, they were merely gleaned for critical incidents: episodes of use in which something goes unexpectedly well or badly (Flanagan 1954, Shattuck & Woods 1994).

Methodological work on usability evaluation sought to develop systematic techniques to ensure cost effectiveness (Bias & Mayhew 1994). Good et al (1986) developed "impact analysis," in which a variety of user performance factors are repeatedly measured throughout the development process to identify those factors that offer the greatest potential impact with respect to given usability specifications. Williges et al (1993) developed a statistical meta-strategy for reducing the factorial space of experimental studies through sequential estimates of relationships among large sets of independent variables.

The cost of user studies became a central issue. Many proposals were made for "heuristic" evaluation, checklist and script-oriented approaches to supplement or even replace direct user testing (Nielsen & Mack 1994, Nielsen & Mollich 1990). Other work examined the empirical parameters for user studies. Nielsen (1994) found that HCI practitioners were able to run only an average of 9 subjects in laboratory usability evaluations. Direct cost-benefit studies determined that only 4–5 experimental subjects found 80% of usability problems (Nielsen 1994, Virzi 1992). [Indeed, there have been recent discussions of "quick and dirty" ethnography (Hughes et al 1994).]

The cost of analytical techniques also became an issue. The assumption made in GOMS that actions are independent and additive rendered GOMS incapable of describing the co-articulation of concurrent activities. These limitations were acknowledged by Card et al (1983); for all these limitations, GOMS was also difficult to apply. John (1990) extended GOMS for tasks involving parallel activities by incorporating the concept of critical path. Kieras (1988) developed a simplified and explicit method for building GOMS analyses through top-down, breadth-first expansion of goals into methods to the level desired. Other simplified cognitive approaches were proposed. "Claims analysis" was suggested to integrate design rationale (see section below) and user modeling in analytic evaluation (Carroll & Rosson 1991). "Cognitive walkthrough" was proposed as heuristic evaluation grounded in simplified cognitive theory (Polson et al 1992). Comparative cost-benefit research is needed and is beginning to appear (Jeffries et al 1991, Karat et al 1992, Nielsen & Phillips 1993).

The term "usability engineering" connotes a practice that is broader and more systematic than is the case. HCI is largely a first-generation field in which various pioneers tend to follow the practices they helped to originate, not always the more eclectic best practices. The pioneer zeitgeist of the field needs to be smoothed and consolidated with increased emphasis on synthesis and integration. For example, although it has been shown that different evaluation methods identify different kinds of usability problems (Jeffries et al 1991), there are no schemes for integrating different types of evaluation data (e.g. quantitative performance data and qualitative protocol data) or for integrating evaluation data collected at different points in the iterative development process. Practitioners do not always differentiate among the variety of evaluation goals they must manage (Carroll & Rosson 1995).

A variety of such integration questions remains at the center of usability engineering. How can user models of the most technical sort (GOMS) be used in participatory design processes? Participatory design and usability engineering are sometimes considered alternative technical approaches (e.g. Monk et al 1993). Can broader user participation improve the cost effectiveness of usability engineering? Can users help to critique or even write usability specifications, heuristic evaluation checklists, walk-through scenarios, and design rationales? Cost-benefit analysis often views benefit fairly narrowly; in the long-term, part of the benefit could be the development of user models and design rationale from the specific results of usability evaluations. Can we develop means of estimating these more profound benefits? Can the development of user models and design rationale be (partially) automated as a by-product of usability engineering activities? How directly can ethnographic analysis guide design work?

Design Rationale

A computer system does not itself elucidate the motivations that initiated its design; the user requirements it was intended to address; the discussions, debates, and negotiations that determined its organization; the reasons for its particular features; the reasons against features it does not have; the weighing of trade-offs; and so forth. Such information can be critical to the variety of stakeholders in a design process: customers, users, servicers, and marketers, as well as designers who want to build upon the system and the ideas it embodies. This information comprises the design rationale of the system (Moran & Carroll 1996).

Approaches to design rationale can be divided into those that describe the design *solution* and those that describe the design *process*. The former approach seeks to position a given design in a larger context of issues and alternative designs. For example, MacLean et al (1991) developed an approach that explicates the design "space" surrounding a given design by enumerating the issues designers identified, the options they considered in responding to these issues, and the criteria they used in weighing the options. Thus, a window interface incorporating a particular technique for scrolling window contents can be located in a design space of other scrolling techniques.

Carroll & Rosson (1991) developed an approach that considers systems to be "embodied" social and behavioral claims about the needs, abilities, and activities of their users. Their approach seeks to articulate the social and behavioral theory implicit in a design. Thus, a programming environment can be seen as embodying a range of claims about what programmers know, what they do, and what they experience, and about the nature of programming tasks and the contexts within which these tasks are carried out.

These approaches make it easy to succinctly summarize the critical usability trade-offs of a particular usage situation, and to link these issues closely with specific features of the computer artifact in use. For example, including animated demonstrations as a self-instruction resource in a programming environment may intrinsically motivate learners and support learning about appropriate goals, but the demonstrations may also place learners in a passive role and suggest goals that are too difficult or nonproductive (e.g. the goal of altering the demonstration itself).

Many process-oriented approaches to design rationale are based on the issue-based information system (IBIS) developed by Rittel (e.g. Rittel & Weber 1973). In this approach, design deliberations are described according to the issues that arise during the design process, the various positions raised in response to the issues, and the arguments for and against each position. Conklin & Yakemovic (1991) showed how such a design rationale could be captured and used in a large commercial project, though they also emphasized

the considerable work involved in coding this rationale. Other process-oriented approaches to rationale emphasize capturing less coded information, such as sketches and design notes, or videotaped self-disclosures by system developers (e.g. Carroll et al 1994).

The emergent nature of the design process makes design rationale both important and difficult. It is a tool for managing the complexity of a process in which everything, including the problem definition, constantly changes. Design rationale can track the consideration of various subproblems, trade-offs, and partial solutions, the reasons for changes and decisions, and the status of assumptions and issues. It can help design teams structure their problem-solving efforts and avoid thrashing. But how much of the design process can be usefully represented, and who will do the work of creating and maintaining this design rationale? Often design rationale is most useful to those who were not even involved with the project when the rationale was created (Grudin 1994). Should every developer have an analyst noting every hypothetical and every train of thought, videotaping every chance encounter in the hall, collecting every sketch and every note?

Current work on design rationale is concerned with assessing and supporting its efficacy. Empirical studies of designers and project teams are investigating how design rationale techniques can be learned and used (e.g. Buckingham Shum 1996). Other work is experimenting with software tools to support the creation of and access to rationales (Conklin & Begeman 1988, Fischer et al 1991, Rosson & Carroll 1995).

Design rationale integrates advances in iterative development and user models. Making the process and outcomes of design more explicit allows iterative development to be more systematic and more manageable. However, it also creates an explicit design representation, a "theory" of the artifact and its use tested by formative and summative evaluation throughout the iterative development process. This theory is not a classic user model; it describes specific situations of use, not purportedly general information processes and cognitive structures. However, this specificity makes it more powerful within the immediate design context, and schemes have been proposed for generalizing such situated theories (e.g. Carroll et al 1992). This integrative role of design rationale exemplifies what Scriven (1967) called "mediated evaluation," an approach in which design analysis of implicit goals and positions on inherent trade-offs among goals is used to guide design evaluation of user performance and experience.

Design rationale can be a language for stakeholders in the design, but these different stakeholders often speak different disciplinary languages, are motivated by different values, and see different technical issues when looking at the "same" design problem. Can everyone use the same design rationale infor-

mation? Will gathering and codifying rationale alter the design process? Will it interfere? Who should have access to the design rationale that is created? There are potential conflicts between privacy rights and access rights of stakeholders. For example, it is useful for developers to be candid about trade-offs they managed, but are users entitled to this information?

Cooperative Activity

A trend to the social has gradually developed in HCI over the past 15 years, a development that has markedly accelerated, deepened, and diversified in the past five (Hiltz & Turoff 1978/1993). This recent trend is actually a nexus of at least four logically independent developments. First, there was a clear consensus by 1990 that the cognitive modeling approach had failed to provide a comprehensive paradigm. Second, many voices suggested a more socially or organizationally oriented approach was required to supplement or replace the cognitive paradigm. Third, the growing technical prominence of HCI attracted a sociopolitical critique of usability as a potential apology for de-skilling and other unpleasant aspects of industrial restructuring. Fourth, new technologies for communication and collaborative activity swept through the computing industry and raised significantly new challenges and opportunities for HCI.

It was somewhat an accident of history that the original foundations of HCI are so strongly in cognitive psychology. The research agenda opened up by the pioneers of software psychology, together with the new opportunities occasioned by personal and distributed computing, attracted a core of cognitive psychologists, who attracted even more. HCI emerged from what in retrospect seems to have been the evangelical heyday of the cognitive paradigm. The initial euphoria with models of ideal performance time was soon displaced by frustration with the limitations of these models, particularly regarding learning in context, error and error recovery, preference, fatigue, work context and collaboration, and individual differences. In 1990 and 1991 the major international conferences in HCI featured panels addressed to the failure of theory (Monk et al 1990, Sutcliffe et al 1991). Recent reformulations of the role of cognitive user modeling position it more eclectically within usability engineering (Nielsen 1993, Preece et al 1994).

In the 1990s, new voices entered the HCI discussion, urging a stronger social and contextual orientation. Anthropologists and sociologists joined what had been largely a cognitive psychology project (Bowker et al 1995, Thomas 1995). European perspectives on technology and work began to penetrate the discourse of what had been a somewhat insular American and British endeavor (Bjerknes et al 1987, Greenbaum & Kyng 1991). Concepts from activity theory, work psychology, and the labor movement became well known (Bødker 1991, Carroll 1991, Ehn 1988, Nardi 1995a). More research attention was directed at understanding situated and distributed cognition (Carroll & Rosson

1992, Norman 1991, Suchman 1995, Zuboff 1988). As part of a larger paradigmatic restructuring of social and behavioral science, traditions that had sought to study individuals in isolation from their contexts, and social phenomena in isolation from individuals, were declining (Knorr-Cetina 1981).

By far the most theoretically rich alternate paradigm is activity theory, derived from the work of Vygotsky (Nardi 1995a, Wertsch 1985). The object of description in this approach is an "activity system," the ensemble of technological factors with social factors and of individual attitudes, experiences, and actions with community practices, traditions, and values. Activity theory emphasizes that these ensembles are inherently contingent and changing, that human activities are mediated and transformed by human creations, such as technologies, and that people make themselves through their use of tools. The tensions and contradictions within an activity system at a given point define a "zone of proximal development" within which people can effect changes (Engestrom 1993). Activity theory shifts attention from characterizing static and individual competencies to characterizing how people can negotiate with the social and technological environment to solve problems and learn, which subsumes many of the issues of situated and distributed cognition.

Kuutti & Arvonen (1992) showed how the design of a medical information system reintegrated the activity system of health professionals in a medical center, a hospital, and throughout the larger community by enabling a shared concept of total patient care, supported by electronic tools and new practices (see also Bødker 1991). Activity theory subsumes many of the methodological issues about participatory design. The user community needs to be receptive to change; the tensions and inconsistencies in the current activity system must be made visible and questioned, that is, brought into the zone of proximal development. It subsumes ethnography: One must study communities of practice in situ and in detail to understand their zone of proximal development. A critical unanswered question about activity theory is whether it can be codified as a (predictive) engineering model (e.g. Blackler 1995), or whether part of the thrust of activity theory is to emphasize aspects of usability that are not susceptible to engineering models.

The social and organizational perspective brought with it a new critique of technology development in HCI. One theme is that different perspectives are inevitable and that conflict must be systemically accommodated and managed. HCI concepts and techniques through the 1980s tended to be directed at a naive notion of engineering optimality. The 1990s brought the view that stakeholder interests often conflict profoundly (e.g. Suchman 1995). Some of the most significant conflicts and misunderstandings are not between users and designers—the hallmark concern of participatory design—but between different constituencies of users (e.g. Blomberg 1995). These conflicts typically

involve power relations between higher- and lower-status employees, between managers and their subordinates, and between teachers and students. Conflicts can of course be salutary for group decision-making (Kling 1991); the point is that they must be addressed.

The second theme of the sociopolitical critique was the potential for usability to unwittingly become a vehicle for de-skilling and disempowering workers. Making someone's work easier reduces the skill required to perform the work; in a given organizational context, it may reduce status, pay, and even job security. The effects can be subtle and complex because the workers themselves may not recognize the potential for such an outcome and may participate in their own de-skilling. This level of analysis received very little attention in the 1980s by the mainstream involved in HCI. Thus, the malaise regarding cognitive user models arose in doubts that the paradigm could achieve more than performance modeling, not in worries that minimizing low-level actions might be fundamentally the wrong goal to pursue. Recent work has explored the dynamic co-design of new technology and new social and organizational structures (Gasser 1986, Kling & Jewett 1994).

New HCI technologies to support meetings and other collaborative activity have encouraged the development of social and contextual analysis. Through the 1980s, electronic mail became an increasingly common communication tool, and other internet tools like newsgroups, multi-user domains, and real-time chat became less arcane. A variety of environments for collaborative problem solving were investigated, including electronic meeting rooms (Nunamaker et al 1991), two-way video "media-spaces" (Bly et al 1993), and three-dimensional "virtual reality" simulations (Leigh et al 1996). The portrait of a solitary user finding and creating information in a computer became background to the portrait of several to many people working together at a variety of times and places. Speculation about new paradigms for education, work, and leisure activity have become rampant in the field and in the culture at large.

The shift of focus in HCI toward cooperative activity raises many methodological issues. As suggested above, the concept of usability becomes more multifarious and therefore susceptible to a multitude of distortions. Agre (1995) discussed a case in which a usability analysis devolved into a study of the malleability of user perceptions about privacy. Grudin (1994) raised the issue that the people who do the work in a group-oriented system may not be the ones who enjoy the benefits. Most of what we know of usability pertains to information creation and management tasks, but in future systems interpersonal communication tasks may become more central than information access. How are various types of cooperative activity affected by various paradigms for computer-mediated communication? Most of our knowledge pertains to the

relatively primitive paradigms based on electronic mail (Hiltz & Turoff 1978/1993, Sproull & Kiesler 1991). What kinds of human communities will be favored by a computer network infrastructure; what kinds will be weakened? Will worldwide interest groups supplant physical neighborhoods as our primary communities? Will our neighborhoods become transformed into local networks (Schuler 1994)?

SYNTHESIZING A DISCIPLINE

In *Sciences of the Artificial,* Simon wrote that "the proper study of mankind is the science of design." The design of human activities and the technologies that support them are a special case of "design" in this broad sense.

HCI has made steady and sometimes dramatic progress as a science of design. It has become a major research area in computer science and the very fulcrum of information technology development. However, the emergence of HCI is ongoing. Perhaps the most impressive current feature of the area is its fragmentation. The paradigmatic consensus of 1970s software psychology and of 1980s cognitive HCI is gone. This is not necessarily bad. Some of the current rifts may help set the agenda for the future. For example, a strong form of contextualism asserts that there is no role for controlled research in HCI (Carroll 1989, Whiteside & Wixon 1987). This is a potentially constructive challenge to the HCI research community. Other rifts are more a matter of mutual neglect. For example, after a brief period of confrontation in the mid-1980s, the proponents of cognitive user modeling have largely disconnected from programmatic discussions of activity theory, and conversely as well.

Mainly, this fragmentation reflects the difficulty of assimilating the great variety of methodologies, theoretical perspectives, driving problems, and people that have become part of HCI. Today's HCI researchers and practitioners are, after all, immigrants from other disciplines and backgrounds. It is not surprising that they often continue to favor what they know how to do. Younger people now entering the field will bring a broader foundation of knowledge and skill, and it is likely that the potential for a broader HCI will be advanced through them.

The recent past suggests three specific technical themes for the near-term future. First, the engineering scope of HCI will continue to broaden beyond user interface interactions. Designing human activities and appropriate technology to support them is likely to become a more integrated endeavor. Second, the success of HCI in generic applications is likely to bolster further domain-specific work, in particular complex application areas. Finally, the impact of HCI on psychology itself, perhaps the strongest fulfillment of Simon's vision of a design science, is likely to progress more concertedly, given

recent developments in ecological and rational psychology, and given a political context that increasingly questions the use of science.

Activity theory emphasizes that the purview of usability is the totality of what people do and experience, and that the diverse facets of usability are interdependent and co-evolve with technology. It has begun to play a role in guiding the envisionment of new technology (e.g. Kuutti & Arvonen 1992). However, it needs to be cultivated as a more comprehensive foundation for the development process: requirements gathering, participatory interaction, software analysis and design, specification and prototyping, system implementation, documentation and instruction design, and usability evaluation. Scenario-based design, a fusion of activity theory with object-oriented software engineering in which narratives of use situations represent software and systems, is one proposal for how this might be achieved (Carroll 1995).

HCI has focused on common denominator technical domains: mass-market applications like word processors and spreadsheets, standard graphical user interfaces, like the Macintosh and Windows, and generic techniques like menu hierarchies, direct manipulation, and programming by demonstration. In the 1980s there were so many studies of user issues in text editing that it was called the "white rat" of HCI. As in the psychology of learning, paradigmatic focus yields a technical coherence, but at the price of confounding eccentricities. In the near future, there is likely to be more HCI research directed at specific technical domains; for example, user interfaces for typesetters (Bødker et al 1987) or petroleum engineers (Brooks 1993), and software tools and environments for teachers and students (Soloway et al 1994) or for scientists and engineers (Gallopoulos et al 1994).

The emergence of HCI in the past two decades illustrates the possibility of psychological inquiry in the context of system development and of progress with fundamental issues joined with engineering design. It demonstrates, for example, how the complex problem-solving of system development is not a straightforward scaling of laboratory-based studies of puzzles. Laboratory situations, after all, are models of real situations in which people are deprived of social and tool resources that constitute those situations. As emphasized by activity theory, human behavior and experience both adapt to and transform social and technological context. Human activities motivate the creation of new tools, but these in turn alter activities, which in time motivates further tools. The context of behavior and experience is changing more rapidly, more incessantly, perhaps more profoundly than ever before in history. HCI design provides an opportunity fundamentally to expand traditional ecological and recent rational approaches to psychology. These approaches consider the environment a causal factor in psychological explanation, but they have generally

ignored the evolution of the environment itself (Anderson 1990, Brunswick 1956, Gibson 1979).

It is exciting to see that the emerging role of social and cognitive science in computer science and the computer industry is far more diverse, pervasive, and critical than imagined in the 1970s. As it has turned out, that role was not to support a received view of design but to help overturn it and help clarify the real nature of design. Nor was that role to recast psychological laws as human-factors guidelines; it was indeed to play a part in driving the evolution of social and cognitive science, and the recognition that computers can be deliberately designed to facilitate human activity and experience only when social and cognitive requirements drive the design process throughout. There is unprecedented potential for interdisciplinary synergy here. Social science has always borne the vision of what human society might become, but it has typically lacked the means to be constructive. Computer science—quite the converse—cannot avoid causing substantial social restructuring. An integrated and effective HCI can be a turning point in both disciplines and, perhaps, in human history.

ACKNOWLEDGMENTS

Some parts of the section on The Emergence of Usability are adapted from Carroll (1992). Some parts of the section on Design Rationale are adapted from the Introduction to Moran & Carroll (1996).

Literature Cited

Agre P. 1995. Conceptions of the user in computer systems design. See Thomas 1995, pp. 67–106

Anderson JR. 1990. *The Adaptive Character of Thought*. Hillsdale, NJ: Erlbaum

Bennett JB. 1984. Managing to meet usability requirements: establishing and meeting software development goals. In *Visual Display Terminals,* ed. J Bennett, D Case, J Sandelin, M Smith, pp. 161–84. Englewood Cliffs, NJ: Prentice-Hall

Bentley R, Hughes JA, Randall D, Rodden T, Sawyer P, et al. 1992. Ethnographically-informed system design for air traffic control. *Proc. CSCW 1994 Conf. Comput.-Support Coop. Work,* Toronto, pp. 123–29. New York: Assoc. Comput. Mach.

Bias RG, Mayhew DJ, eds. 1994. *Cost-Justifying Usability*. Boston: Academic

Bjerknes G, Ehn P, Kyng M, eds. 1987. *Computers and Democracy: A Scandinavian Challenge*. Brookfield, VT: Avebury

Blackler F. 1995. Activity theory, CSCW and organizations. See Monk & Gilbert 1995, pp. 223–48

Blomberg JL. 1995. Ethnography: aligning field studies of work and system design. See Monk & Gilbert 1995, pp. 175–97

Bly S, Harrison S, Irwin S. 1993. Media spaces: bringing people together in a video, audio, and computing environment. *Commun. Assoc. Comput. Mach.* 36(1):28–47

Bødker S. 1991. *Through the Interface: A Human Activity Approach to User Interface Design*. Hillsdale, NJ: Erlbaum

Bødker S, Ehn P, Kammersgaard J, Kyng M, Sundblad Y. 1987. A utopian experience. See Bjerknes et al 1987, pp. 251–78

Bowker G, Star SL, Gasser L, Turner B, eds. 1995. *Social Science Research, Technical Systems and Cooperative Work*. Mahwah, NJ: Erlbaum

Brooks FP. 1975/1995. *The Mythical Man-Month: Essays on Software Engineering*. Reading, MA: Addison-Wesley

Brooks R. 1993. The case for the specialized interface. *IEEE Softw.* 10(2):86–88

Brunswick E. 1956. *Perception and the Representative Design of Psychological Experiments*. Berkeley: Univ. Calif. Press

Buckingham Shum S. 1996. Analyzing the usability of a design rationale notation. In *Design Rationale: Concepts, Techniques, and Use,* ed. TP Moran, JM Carroll, pp. 185–215. Mahwah, NJ: Erlbaum

Butler KA. 1985. Connecting theory and practice: a case study of achieving usability goals. *Proc. CHI'85 Hum. Factors Comput. Syst., San Francisco,* pp. 85–88. New York: Assoc. Comput. Mach.

Card SK, Moran TP, Newell A. 1983. *The Psychology of Human-Computer Interaction*. Hillsdale, NJ: Erlbaum

Carroll JM. 1985. *What's in a Name? An Essay in the Psychology of Reference*. New York: Freeman

Carroll JM. 1989. Evaluation, description and invention: paradigms for human-computer interaction. In *Advances in Computers,* ed. M Yovits, 29:47–77. San Diego: Academic

Carroll JM. 1990. *The Nurnberg Funnel: Designing Minimalist Instruction for Practical Computer Skill*. Cambridge, MA: MIT Press

Carroll JM, ed. 1991. *Designing Interaction: Psychology at the Human-Computer Interface*. New York: Cambridge Univ. Press

Carroll JM. 1992. Creating a design science of human-computer interaction. In *Future Tendencies in Computer Science, Control, and Applied Mathematics: 25th Anniversary of INRIA,* ed. A Bensoussan, J-P Verjus, pp. 205–15. New York: Springer-Verlag

Carroll JM, Alpert SR, Karat J, Van Deusen M, Rosson MB. 1994. Capturing design history and rationale in multimedia narratives. In *Proc. CHI'94: Hum. Factors Comput. Syst., Boston,* pp. 192–97. New York: Assoc. Comput. Mach./Addison-Wesley

Carroll JM, Campbell RL. 1986. Softening up hard science: reply to Newell and Card. *Hum.-Comput. Interact.* 2:227–49

Carroll JM, Carrithers C. 1984. Blocking learner errors in a training wheels system. *Hum. Factors* 26(4):377–89

Carroll JM, Mack RL, Kellogg WA. 1988. Interface metaphors and user interface design. See Helander 1988, pp. 67–85

Carroll JM, Rosson MB. 1985. Usability specifications as a tool in iterative development. In *Advances in Human-Computer Interaction,* ed. HR Hartson, 1:1–28. Norwood, NJ: Ablex

Carroll JM, Rosson MB. 1991. Deliberated evolution: stalking the View Matcher in design space. *Hum.-Comput. Interact.* 6: 281–318

Carroll JM, Rosson MB. 1992. Getting around the task-artifact cycle: how to make claims and design by scenario. *Assoc. Comput. Mach. Trans. Inf. Syst.* 10:181–212

Carroll JM, Rosson MB. 1995. Managing evaluation goals for training. *Commun. Assoc. Comput. Mach.* 38(7):40–48

Carroll JM, Singley MK, Rosson MB. 1992. Integrating theory development with design evaluation. *Behav. Inf. Technol.* 11: 247–255

Carroll JM, Thomas JC. 1982. Metaphor and the cognitive representation of computing systems. *IEEE Trans. Syst., Man Cybern.* 12:107–16

Carroll JM, Thomas JC, Malhotra A. 1979. A clinical-experimental analysis of design problem solving. *Design Stud.* 1:84–92

Conklin J, Begeman M. 1988. gIBIS: a hypertext tool for exploratory policy discussion. *Assoc. Comput. Mach. Trans. Off. Inf. Syst.* 6(4):303–31

Conklin J, Yakemovic KCB. 1991. A process-oriented approach to design rationale. *Hum.-Comput. Interact.* 6:357–91

Curtis B, Krasner H, Iscoe N. 1988. A field study of the software design process for large systems. *Commun. Assoc. Comput. Mach.* 31:1268–87

Cypher A, Halbert D, Kurlander D, Lieberman H, Maulsby D, et al, eds. 1993. *Watch What I Do: Programming by Demonstration*. Cambridge, MA: MIT Press

deGroot AD. 1965. *Thought and Choice in Chess*. Paris: Mouton

Denning PJ, Comer DE, Gries D, Mulder MC, Tucker AB, et al. 1989. Computing as a discipline. *Commun. Assoc. Comput. Mach.* 32:9–23

Dix A, Finlay J, Abowd G, Beale R. 1993. *Human-Computer Interaction*. Englewood Cliffs, NJ: Prentice-Hall

Douglas SA, Moran TP. 1983. Learning text editor semantics by analogy. *Proc. CHI'83 Conf. Hum. Factors Comput. Syst., Boston,* pp. 207–11. New York: Assoc. Comput. Mach.

Dreyfuss H. 1955. *Designing for People*. New York: Simon & Schuster

Ehn P. 1988. *Work-Oriented Design of Computer Artifacts*. Stockholm: Arbetslivscentrum

Engestrom Y. 1993. Developmental work research: reconstructing expertise through expansive learning. In *Human Jobs and Computer Interfaces,* ed. M Nurminen, G Weir. Amsterdam: Elsevier

Ericsson KA, Simon HA. 1985. *Protocol Analysis: Verbal Reports as Data*. Cambridge, MA: MIT Press

Esper EA. 1925. A technique for the experimental investigation of associative interference in artificial linguistic material. *Lang. Monogr.* 1:1–47

Fischer G, Lemke AC, McCall R, Morch AI. 1991. Making argumentation serve design. *Hum.-Comput. Interact.* 6(3/4):393–419

Flanagan JC. 1954. The critical incident technique. *Psychol. Bull.* 51:28–35

Furnas G, Landauer TK, Gomez L, Dumais S. 1983. Statistical semantics: analysis of the potential performance of keyword information systems. *Bell Syst. Tech. J.* 62: 1753–806

Gallopoulos E, Houstis E, Rice JR. 1994. Computer as thinker/doer: problem-solving environments for computational science. *IEEE Comput. Sci. Eng.* 1(1):11–23

Gasser L. 1986. The integration of computing and routine work. *Assoc. Comput. Mach. Trans. Off. Inf. Syst.* 4(3):205–25

Gibson JJ. 1979. *The Ecological Approach to Visual Perception*. Boston: Houghton Mifflin

Good M, Spine TM, Whiteside J, George P. 1986. User-derived impact analysis as a tool for usability engineering. *Proc. CHI'86 Conf. Hum. Factors Comput. Syst., Boston,* pp. 241–46. New York: Assoc. Comput. Mach.

Gould JD, Boies SJ. 1983. Human factors challenges in creating a principal support office system: the speech filing approach. *Assoc. Comput. Mach. Trans. Off. Inf. Syst.* 1: 273–98

Gray WD, John BE, Atwood ME. 1992. The précis of project Ernestine, or, An overview of a validation of GOMS. *Proc. CHI'92 Conf. Hum. Factors Comput. Syst., Monterey, CA,* pp. 307–12. New York: Assoc. Comput. Mach.

Greatbatch D, Heath C, Luff P, Campion P. 1995. Conversation analysis: human-computer interaction and the general practice consultation. See Monk & Gilbert 1995, pp. 199–222

Greenbaum J, Kyng M, eds. 1991. *Design at Work: Cooperative Design of Computer Systems*. Hillsdale, NJ: Erlbaum

Grudin J. 1989. The case against user interface consistency. *Commun. Assoc. Comput. Mach.* 32:1164–73

Grudin J. 1994. Groupware and social dynamics: eight challenges for developers. *Commun. Assoc. Comput. Mach.* 37(1):92–105

Helander M, ed. 1988. *Handbook of Human-Computer Interaction*. Amsterdam: North-Holland

Hiltz SR, Turoff M. 1978/1993. *The Network Nation: Human Communication via Computer*. Cambridge: MIT Press

Hughes J, King V, Rodden T, Anderson H. 1994. Moving out from the control room: ethnography in system design. *Proc. CSCW'94 Conf. Comput.-Support. Coop. Work, Chapel Hill, NC,* pp. 429–39. New York: Assoc. Comput. Mach.

Hutchins E, Hollan J, Norman DA. 1986. Direct manipulation interfaces. See Norman & Draper 1986, pp. 87–124

Jeffries RJ, Miller JR, Wharton C, Uyeda KM. 1991. User interface evaluation in the real world: a comparison of four techniques. *Proc. CHI'91 Hum. Factors Comput. Syst., New Orleans,* pp. 119–24. New York: Assoc. Comput. Mach.

John BE. 1990. Extensions of GOMS analyses to expert performance requiring perception of dynamic visual and auditory information. *Proc. CHI'90 Hum. Factors Comput. Syst., Seattle,* pp. 107–15. New York: Assoc. Comput. Mach.

Karat CM, Campbell RL, Fiegel T. 1992. Comparison of empirical testing and walkthrough methods in user interface evaluation. *Proc. CHI'92 Hum. Factors Comput. Syst., Monterrey, CA,* pp. 397–404. New York: Assoc. Comput. Mach.

Kieras DE. 1988. Towards a practical GOMS model methodology for user interface design. See Helander 1988, pp. 135–57

Kling R. 1991. Cooperation, coordination and control in computer-supported work. *Commun. Assoc. Comput. Mach.* 34(12):83–88

Kling R, Jewett T. 1994. The social design of worklife with computers and networks: a natural systems perspective. In *Advances in Computers,* ed. M Yovits, 39:239–93. Orlando: Academic

Knorr-Cetina KD. 1981. The micro-sociological challenge of macro-sociology: towards a reconstruction of social theory and methodology. In *Advances in Social Theory and Methodology: Towards an Integration of Micro- and Macro-Sociologies,* ed. KD Knorr-Cetina, A Cicourel. London: Routledge & Kegan Paul

Kuhn S, Muller MJ, eds. 1993. Special section on participatory design. *Commun. Assoc. Comput. Mach.* 36(4):24–103

Kuutti K, Arvonen T. 1992. Identifying CSCW applications by means of activity theory concepts: a case example. *Proc. CSCW'92 Comput.-Support. Coop. Work, Toronto,* pp. 233–40. New York: Assoc. Comput. Mach.

Leigh J, Johnson AE, Vasilakis CA, DeFanti TA. 1996. Multi-perspective collaborative design in persistent networked virtual environments. *Proc. IEEE Virtual Reality Annu. Int. Symp., VRAIS 96,* Santa Clara, CA. In press

Lewis CH, Norman DA. 1986. Designing for error. See Norman & Draper 1986, pp. 411–32

Mack RL, Lewis CH, Carroll JM. 1983. Learning to use office systems: problems and prospects. *Assoc. Comput. Mach. Trans. Off. Inf. Syst.* 1:254–71

MacLean A, Young RM, Bellotti VME, Moran TP. 1991. Questions, options, and criteria: elements of design space analysis. *Hum.-Comput. Interact.* 6:201–50

Malhotra A, Thomas JC, Carroll JM, Miller LA. 1980. Cognitive processes in design. *Int. J. Man-Mach. Interact.* 12:119–40

Miller LA. 1974. Programming by nonprogrammers. *Int. J. Man-Mach. Stud.* 6: 237–60

Monk AM, Carroll JM, Harrison M, Long J, Young RM. 1990. New approaches to theory in HCI: How should we judge their acceptability? *Proc. INTERACT'90 Hum.-Comput. Interact., Cambridge,* pp. 1055–58. Amsterdam: North-Holland

Monk AM, Gilbert N, eds. 1995. *Perspectives on HCI: Diverse Approaches*. London: Academic

Monk AM, Nardi B, Gilbert N, Mantei M, McCarthy J. 1993. Mixing oil and water? Ethnography versus experimental psychology in the study of computer-mediated communication. *Proc. INTERCHI'93 Hum. Factors Comput. Syst., Amsterdam,* pp. 3–6. New York: Assoc. Comput. Mach.

Moran TP. 1983. Getting into a system: external-internal task mapping analysis. *Proc. CHI'83 Hum. Factors Comput. Syst., Boston,* pp. 45–49. New York: Assoc. Comput. Mach.

Moran TP, Carroll JM, eds. 1996. *Design Rationale: Concepts, Techniques, and Use*. Mahwah, NJ: Erlbaum

Nardi BA, ed. 1995a. *Context and Consciousness: Activity Theory and Human-Computer Interaction*. Cambridge, MA: MIT Press

Nardi BA. 1995b. Studying context: a comparison of activity theory, situated action models and distributed cognition. See Nardi 1995a, pp. 69–102

Newell A, Card SK. 1985. The prospects for psychological science in Human-Computer Interaction. *Hum.-Comput. Interact.* 1: 209–42

Newell A, Simon HA. 1972. *Human Problem Solving*. Englewood Cliffs, NJ: Prentice-Hall

Nielsen J, ed. 1989. *Coordinating User Interfaces for Consistency*. New York: Academic

Nielsen J. 1993. *Usability Engineering*. Boston: Academic

Nielsen J. 1994. Estimating the number of subjects needed for a thinking aloud test. *Int. J. Hum.-Comput. Stud.* 41:385–97

Nielsen J, Mack RL, eds. 1994. *Usability Inspection Methods*. New York: Wiley

Nielsen J, Mollich R. 1990. Heuristic evaluation of user interfaces. *Proc. CHI'90 Conf. Hum. Factors Comput. Syst., Seattle,* pp. 249–56. New York: Assoc. Comput. Mach.

Nielsen J, Phillips VL. 1993. Estimating the relative usability of two interfaces: heuristic, formal and empirical methods compared. *Proc. INTERCHI'93 Conf. Hum. Factors Comput. Syst., Amsterdam,* pp. 214–21. New York: Assoc. Comput. Mach.

Norman DA. 1991. Cognitive artifacts. See Carroll 1991, pp. 17–38

Norman DA, Draper SW, eds. 1986. *User Centered System Design*. Hillsdale, NJ: Erlbaum

Nunamaker J, Dennis A, Valacich J, Vogel D, George J. 1991. Electronic meeting systems to support group work. *Commun. Assoc. Comput. Mach.* 34(7):40–61

Nygaard K. 1979. The iron and metal project: trade union participation. In *Computers Dividing Man and Work: Recent Scandinavian Research on Planning and Computers from a Trade Union Perspective,* ed. A Sandberg, p. 98. Malmö: Arbetslivscentrum

Pava CHP. 1983. *Managing New Office Technology: An Organizational Strategy*. New York: Free Press

Payne SJ, Green TRG. 1989. Task-Action Grammars: the model and its developments. In *Task Analysis for Human-Computer Interaction,* ed. D Diaper, pp. 75–107. Chichester: Ellis Horwood

Payne SJ, Squibb H, Howes A. 1990. The nature of device models: the yoked state space hypothesis and some experiments with text editors. *Hum.-Comput. Interact.* 5:415–44

Polson PG, Lewis CH, Rieman J, Wharton C. 1992. Cognitive walkthroughs: a method for theory-based evaluation of user interfaces. *Int. J. Man-Mach. Stud.* 36:741–73

Preece J, Rogers Y, Sharp H, Benyon D, Holland S, Carey T. 1994. *Human-Computer Interaction*. Reading, MA: Addison-Wesley

Rittel H, Webber M. 1973. Dilemmas in a general theory of planning. *Policy Sci.* 4: 155–69

Rosson MB, Carroll JM. 1995. Narrowing the specification-implementation gap in scenario-based design. In *Scenario-Based Design: Envisioning Work and Technology in System Development,* ed. JM Carroll, pp. 247–78. New York: Wiley

Royce WW. 1970. Managing the development of large software systems: concepts and techniques. *Proc. West. Electr. Show Conv., WESTCON, Los Angeles,* pp. (A/1)1-9. Reprinted in *Proc. Int. Conf. Softw. Eng., 11th, Pittsburgh,* May 1989, pp. 328–38

Schuler D. 1994. Community networks: building a new participatory medium. *Commun. Assoc. Comput. Mach.* 37(1):39–51

Schuler D, Namioka A, eds. 1993. *Participatory Design: Principles and Practices*. Hillsdale, NJ: Erlbaum

Scriven M. 1967. The methodology of evalu-

ation. In *Perspectives of Curriculum Evaluation*, ed. R Tyler, R Gagne, M Scriven, pp. 39–83. Chicago: Rand McNally

Shattuck L, Woods D. 1994. The critical incident technique: 40 years later. *Proc. Hum. Factors Ergon. Soc. Annu. Meet., 38th, Nashville,* pp. 1080–84. Santa Monica, CA**:** Hum. Factors Ergon. Soc.

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 Comput*. 16(8):57–62

Shneiderman B, Mayer R, McKay D, Heller P. 1977. Experimental investigations of the utility of detailed flowcharts in programming. *Commun. Assoc. Comput. Mach.* 20: 373–81

Sime ME, Green TRG, Guest DJ. 1973. Psychological evaluation of two conditional constructions used in computer languages. *Int. J. Man-Mach. Stud.* 5:105–13

Simon HA. 1969. *The Sciences of the Artificial*. Cambridge: MIT Press

Smith DC, Irby C, Kimball R, Verplank B, Harslem E. 1982. Designing the Star user interface. *Byte* 7(4):242–82

Soloway E, Guzdial M, Hay KE. 1994. Learner-centered design: the challenge for HCI in the 21st Century. *Interactions* 1(2): 36–48

Sproull L, Kiesler S. 1991. *Connections: New Ways of Working in A Networked Organization*. Cambridge: MIT Press

Suchman LA. 1987. *Plans and Situated Actions: The Problem of Human-Machine Communication*. New York: Cambridge Univ. Press

Suchman LA, ed. 1995. Special section on representations of work. *Commun. Assoc. Comput. Mach.* 38(9):33–68

Sukaviriya P, Foley JD, Griffith T. 1993. A second-generation user interface design environment: the model and runtime architecture. *Proc. INTERCHI'93 Conf. Hum. Factors Comput. Syst., Amsterdam,* pp. 375–82. New York: Assoc. Comput. Mach.

Sutcliffe A, Carroll JM, Young RM, Long J. 1991. HCI theory on trial. *Proc. INTERCHI'93 Conf. Hum. Factors Comput. Syst., Amsterdam,* pp. 375–82. New York: Assoc. Comput. Mach.

Szekely P, Luo P, Neches R. 1993. Beyond interface builders: model-based interface tools. *Proc. CHI'91 Conf. Hum. Factors Comput. Syst., New Orleans,* pp. 399–401. New York: Assoc. Comput. Mach.

Tanner PP, Buxton WAS. 1985. Some issues in future user interface management system (UIMS) development. In *User Interface Management Systems: Proc. Workshop User Interface Management Syst., Seehiem, Ger., 1983,* ed. GE Pfaff, 67–79. New York: Springer-Verlag

Thomas PJ, ed. 1995. *The Social and Interactional Dimensions of Human-Computer Interact*. New York: Cambridge Univ. Press

Tucker AB, ed. 1997. *The Handbook of Computer Science and Engineering*. Boca Raton, FL: CRC Press

Tucker AB, Turner AJ. 1991. A summary of the ACM/IEEE-CS Joint Curriculum Task Force Report: computing Curricula 1991. *Commun. Assoc. Comput. Mach.* 34:68–84

Virzi RA. 1992. Refining the test phase of usability evaluation: how many subjects is enough? *Hum. Factors* 34:457–68

Wasserman AI, Shewmake DT. 1982. Rapid prototyping of interactive information systems. *Assoc. Comput. Mach. Softw. Eng. Notes* 7:171–80

Wasserman AI, Shewmake DT. 1985. The role of prototypes in the User Software Engineering (USE) methodology. In *Advances in Human-Computer Interaction,* ed. HR Hartson, 1:191–209. Norwood, NJ: Ablex

Weissman L. 1974. *A methodology for studying the psychological complexity of computer programs*. PhD thesis. Univ. Toronto, Toronto

Wertsch J. 1985. *Vygotsky and the Social Formation of Mind*. Cambridge, MA: Harvard Univ. Press

Whiteside J, Bennett J, Holtzblatt K. 1988. Usability engineering: our experience and evolution. See Helander 1988, pp. 791–817

Whiteside J, Wixon D. 1987. Improving human-computer interaction: a quest for cognitive science. In *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction,* ed. JM Carroll, pp. 337–52. Cambridge, MA: Bradford/MIT Press

Williges RC, Williges BH, Han SH. 1993. Sequential experimentation in human-computer interface design. In *Advances in Human-Computer Interaction,* ed. HR Hartson, D Hix, 4:1–30. Norwood, NJ: Ablex

Wixon D, Holtzblatt K, Knox S. 1990. Contextual design: an emergent view of system design. In *Proc. CHI '90: Hum. Factors Comput. Sys.,* Seattle, pp. 329–36. New York: Assoc. Comput. Mach.

Wright RB, Converse SA. 1992. Method bias and concurrent verbal protocol in software usability testing. *Proc. Hum. Factors Ergon. Soc. Annu. Meet., 36th, Atlanta,* pp. 891–912. Santa Monica, CA**:** Hum. Factors Ergon. Soc.

Zuboff Z. 1988. *In the Age of the Smart Machine: The Future of Work and Power*. New York: Basic Books