Int. J. Human-Computer Studies (1997) **46,** 501–522

Human–computer interaction: psychology as a science of design

John M. Carroll

Computer Science Department and Center for Human-Computer Interaction, Virginia Tech, Blacksburg, VA 24061-0106 USA. email: carroll@cs.vt.edu

(Received 8 November 1996 and accepted 6 December 1996)

Human–computer interaction (HCI) is the area of intersection between psychology and the social sciences, on the one hand, and computer science and technology, on the other. HCI researchers analyse and design-specific user-interface technologies (e.g. three-dimensional pointing devices, interactive video). They study and improve the processes of technology development (e.g. usability evaluation, design rationale). They develop and evaluate new applications of technology (e.g. computer conferencing, software design environments). Through the past two decades, HCI has progressively integrated its scientific concerns with the engineering goal of improving the *usability* of computer systems and applications, thus establishing a body of technical knowledge and methodology. HCI continues to provide a challenging test domain for applying and developing psychology and social science in the context of technology development and use.

1. The emergence of usability

Human–computer interaction (HCI) has emerged as a focal area of both computer science research and development and of applied psychology. Some of the reasons for its success are straightforwardly technical: HCI has evoked many difficult problems and elegant solutions in the recent history of computing (for example, in work on direct manipulation interfaces, user-interface management systems, task-oriented help and instruction and computer-supported collaborative work). Other reasons are broadly cultural: the province of HCI covers the realm of opinions and ideas that the non-specialist public has of computer and information technology and the impact that technology has on their lives, in this sense, it is the visible part of computer science. The most recent reasons are commercial: as the underlying technologies of computing become commodities, inscribed on generic chips, the non-commodity value of computer products resides in applications and user interfaces, that is, in HCI.

The area has evolved rapidly in the past two decades as it has struggled to define for itself intellectual substance and practical technique, that is to say, aspired for both a basis in science and a utility in system and software development. In this paper, I review the history of HCI as steps toward a science of design. My touchstone is Simon's (1969) provocative book *The Sciences of the Artificial*. The book pre-dates 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 for charting its continuing development.

1071-5819/97/040501 + 22$25.00/0/hc960101

Early in the book, Simon discusses 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 is pursuing a relatively simple goal and accommodating to whatever the beach presents. The external world, including the technology humans 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 laments 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.

1.1. 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 distinctive methodological axioms: The first was to assume 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 was to assume two central roles for psychology within this context: (a) to produce a general description of human beings interacting with systems and software, a description which could be synthesized as a guideline for developers and (b) to verify directly 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, Green & Guest, 1973), classifying errors people make 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, Mayer, McKay & Heller, 1977). This work inspired many industrial human factor groups to expand the scope of their responsibilities toward support for programming groups and the usability of software.

But the basic axioms of software psychology proved to be problematic. The waterfall idealization of design work is both infeasible and ineffective (e.g. Brooks, 1975/1995). It is only observed when enforced, and 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 its 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 contrasts (organized vs. disorganized menus, structured vs. 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 the 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 upon heroic developers.

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

1.2. 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. Through the course of this process, they sometimes radically reformulate the fundamental goals and constraints of the problem. But this is not a chaos; it is a highly involuted and highly structured process of problem discovery and clarification in the context of unbounded complexity (e.g. Carroll, Thomas & Malhotra, 1979; Malhotra, Thomas, Carroll & Miller, 1980; Curtis, Krasner & Iscoe, 1988).

The leading idea is that designers often need to *do* design in order to adequately understand design problems. One prominent empirical case was Brooks' (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. 116–123). This was a striking lesson to draw, and carried with it many implications. For example, formal and comprehensive planning and specification aids (like detailed flowcharts) will have limited use 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' approach institutionalizes an accommodation to designers' propensity for concrete, incremental reasoning and testing. It incorporates four central ideas: (1) early prototyping, (2) the involvement of real users, (3) introduction of new functions through familiar "survival forms" and (4) many cycles of design iteration. Dreyfuss pushes 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' design work on airplane interiors for Lockheed. Dreyfuss sent two associates back and forth across the US on commercial flights to monitor 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 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 h: 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.

Through the decade of 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 (Nygaard, 1979; Gould & Boies, 1983; Pava, 1983); it emphasized user-interface metaphors for presenting novel functionality through familiar concepts (Carroll & Thomas, 1982; Smith, Irby, Kimball, Verplank & Harslem, 1982); and it made "rapid prototyping" a standard system development methodology (Wasserman & Shewmake, 1982, 1985).

Iterative development shifted the focus of usability evaluation from the summative to the 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 a process of continual redesign. "Thinking aloud" had been pioneered by deGroot (1965) and Newell and Simon (1972) in the study of puzzles and games, and 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, Lewis & Carroll, 1983; Wright & Converse, 1992).

1.3. USER MODELS

The second problem-area bequeathed to HCI by software psychology was the characterization of a robust science base that could underwrite system development. The cornerstone in this effort was the GOMS project of Card, Moran and Newell (1983). GOMS, which stands for Goals, Operators, Methods and Selection rules, provided a framework for analysing systematically 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, John & Atwood, 1992).

One of the salient limitations of GOMS models is that they do not describe learning, yet the challenges faced by new users getting started 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, Squibb & Howes, 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" (vs., 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, Landauer, Gomez & Dumais, 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" that was gradually articulated through the 1980s (Shneiderman, 1983; Hutchins, Hollan & Norman, 1986). It also led to theories of user-interface metaphor (Carroll, Mack & Kellogg, 1988).

A second limitation of the early GOMS-style cognitive models is that they did not address problem-solving and error. Studies showed that new users spent a third to half of their time in error recovery (e.g. Mack *et al*., 1983), but more significantly, these studies showed that often the dispositions that make people good sense makers also cause characteristic learner problems (Carroll, 1990): people want to learn by doing, but this inclines them to jump around opportunistically in sometimes-brittle learning sequences. They want to reason things out and construct their own understandings, but they do not always plan and they often draw incorrect inferences. They try to engage and extend their 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", improvizing, hypothesizing, trying to make sense of a very complex environment. It led to emphasis on designing for learning-by-doing and for error (Lewis & Norman, 1986; Carroll, 1990). For example, Carroll and 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 permitted to try again without penalty. This kind of design supports sensemaking, but not necessarily efficient performance. To some extent, the active user was an alternative to the GOMS conception of the user as an information processor.

2. User-centered system development

At the inception 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 of10 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 enumerated 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 IEEE (Institute of Electrical and Electronic Engineers) recommended the inclusion of HCI as a common requirement in computer science programs (Tucker & Turner, 1991). And HCI was included as one of 10 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, Finlay, Abowd & Beale, 1993; Preece, Rogers, Sharp, Benyon, Holland & Carey, 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 in installation and user support. There has been an obvious trend for HCI specialists to be promoted into project management. None of these trends impugn 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 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 on usability engineering, design rationale and cooperative activity.

2.1. 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 through the 1980s (Whiteside, Bennett & Holtzblatt, 1988; Nielsen, 1993). 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 10 min with fewer than three errors after 30 min training". Usability specifications are now a 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 emphasizing "low-tech", cooperative activities to facilitate collaboration between users, who bring expertise regarding the work situation, and developers, who bring expertise regarding 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 come into existence.

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, Holtzblatt & Knox, 1990). It overturned the laboratory bias inherited by HCI from cognitive psychology, 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. But conversely, field studies cannot reveal the perspectives and insights users bring to the development process as design team members.

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, 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 grain behavioral structure (Greatbatch, Heath, Luff & Campion, 1995). The commitment to revelatory interpretation of situated behavior can entail a behaviorism that undervalues what people experience and report (Nardi, 1995*b*).

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, Foley & Griffith, 1993; Szekely, Luo & Neches, 1993). Concerns about ease of *implementation* motivated a family of prototyping tools based on the premise that user-interface software could directly be 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, Moran & Newell, 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 analysed 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, Spine, Whiteside and George (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 the given usability specifications. Williges, Williges and Han (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 usability inspections, checklist and script-oriented approaches to supplement or even replace direct user testing (Nielsen & Molich, 1990; Nielsen & Mack, 1994). Other work sought to specify and minimize empirical parameters for user studies. Nielsen (1994) found that HCI practitioners were only able to run an average of nine subjects in laboratory usability evaluations. Direct cost–benefit studies determined that only four to five experimental subjects could find 80% of usability problems (Nielsen, 1994; Virzi, 1992). (Indeed, there have been recent discussions of "quick and dirty" ethnography; Hughes, King, Rodden & Anderson, 1994.)

The cost effectiveness of analytical techniques also became an issue. The assumption made by GOMS that actions are independent and additive rendered the model incapable of describing the co-articulation of activities. John (1990) extended GOMS for tasks involving concurrent activities by incorporating the concept of critical path. GOMS was also acknowledged to be difficult to apply (Newell & Card, 1985). Kieras (1988) developed a simplified and explicit method for building GOMS analyses through top-down, breadth-first expansion of goals into methods, expanded to the level desired.

Other cost effective cognitive modeling approaches were proposed. Claims analysis was suggested to integrate design rationale (see the following subsection) and scenario-based user modeling (Carroll & Rosson, 1991). Cognitive walkthrough was proposed as a usability oriented development of traditional program walkthroughs (Polson, Lewis, Rieman & Wharton, 1992). The focus on cost effectiveness has led in some cases to potentially regressive simplifications. For example, some inspection methods take no account of task context, implicitly assuming that the features of an interface or application can be interpreted without a context of use (Nielsen & Molich, 1990; Nielsen & Mack, 1994). Comparative cost–benefit research is needed and is beginning to appear (Jeffries, Miller, Wharton & Uyeda, 1991; Karat, Campbell & Fiegel, 1992; Nielsen & Phillips, 1993).

The term "usability engineering" connotes a practice that is broader and more systematic than is currently 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 remain at the center of usability engineering. How can user models of the most technical sort be used in participatory design processes? Participatory design and usability engineering are sometimes considered *alternative* technical approaches (e.g. Monk, Nardi, Gilbert, Mantei & McCarthy, 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?

2.2. 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 tradeoffs, 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 type of approach seeks to position a given design in a larger context of issues and alternative designs. For example, MacLean, Young, Belloti and Moran (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 and 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 to rationale 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 summarize succinctly the critical usability tradeoffs pertaining to 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 non-productive (e.g. the goal of altering the demonstration itself).

Most process-oriented approaches to design rationale are based on the "issue-based information system" (IBIS) framework, developed by Rittel (e.g. Rittel & Weber, 1973). In this approach, design deliberations are described in terms of the issues that arise in the course of the design process, the various positions raised in response to the issues, and the arguments for and against each position. Conklin and Yakemovic (1991) showed how such design rationale could be captured and used in a large commercial project, though they also emphasize 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, Alpert, Karat, Van Deusen & Rosson, 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 sub-problems, tradeoffs 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, Lemke, McCall & Morch, 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. But 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, Singley & Rosson, 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 underlying tradeoffs 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 information? 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 among themselves about tradeoffs they managed; are users entitled to this information?

2.3. 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 that 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 socio-political 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 heydays 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, Carroll, Harrison, Long & Young, 1990; Sutcliffe, Carroll, Young & Long, 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, Star, Gasser & Turner, 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, Ehn & Kyng, 1987; Greenbaum & Kyng, 1991). Concepts from activity theory, work psychology and the labor movement became well-known (Ehn, 1988; Bødker, 1991; Carroll, 1991; Nardi, 1995*a*). More research attention was directed at understanding situated and distributed cognition (Zuboff, 1988; Norman, 1991; Carroll & Rosson, 1992; Suchman, 1995). All this should be seen 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 (Wertsch, 1985; Nardi, 1995*a*). 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 in time define a "zone of proximal development" within which people can effect changes (Engestrom, 1993). Activity theory shifts attention from characterizing static and individual competencies toward 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 and 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 of 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 still-open question about activity theory is whether it can be codified as an (predictive) engineering

model (e.g. Blackler, 1995), or whether part of the thrust of activity theory is to emphasize and address 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 systematically 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, 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 socio-political 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, 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 of 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, Dennis, Valacich, Vogel & George, 1991), two-way video "media-spaces" (Bly, Harrison & Irwin, 1993), and three-dimensional "virtual reality" simulations (Leigh, Johnson, Vasilakis & DeFanti, 1996). The portrait of a solitary user finding and creating information in a personal computer became background to the portrait of several 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) discusses a case in which a usability analysis devolved into a study of the malleability of user perceptions about privacy. Grudin (1994) raises 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 inter-personal 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 as local networks (Schuler, 1996)?

3. Synthesizing a discipline

In *Sciences of the Artificial*, Simon wrote, "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. But it is perhaps the frontier for a science of design. The science of design is about the iterative development process, about models of users and their tasks, about users and designers understanding one another and working together to create possibilities. The science of design involves describing designs and the design process, through rationale and history, to understand better what was done, why it was done and how it might be improved. It involves developing an engineering practice to integrate problem understanding, solution envisionment, implementation, measurement, prediction and technical management. It involves understanding and managing human activity systems to satisfy the myriad goals and preferences of the organizations and individuals that comprise those systems and the constraints and affordances of their artifacts.

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 (Whiteside & Wixon, 1987; Carroll, 1989). 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 traditional cognitive user modeling have largely disconnected from programmatic discussions of activity theory, and conversely as well.

To a considerable extent, 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 cultural context which increasingly is asking what science is for.

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, 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, Ehn, Kammersgard, Kyng & Sundblad, 1987) or for petroleum engineers (Brooks, 1993), and software tools and environments for teachers and students (Soloway, Guzdial & Hay, 1994) or for scientists and engineers (Gallopoulos, Houstis & Rice, 1994).

The emergence of HCI in the past two decades illustrates the possibility of psychological inquiry in the context of system development, 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. Indeed, as emphasized by activity theory, human behavior and experience both adapts to and transforms 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 to expand fundamentally traditional ecological and recent rational approaches to psychology. These approaches take the environment as 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 to 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.

An earlier version of this paper appeared in Annual Review of Psychology (1997), **48**, 61–83. Palo Alto, CA: Annual Reviews.

References

AGRE, P. (1995). Conceptions of the user in computer systems design. In P. J. THOMAS, Ed. *The Social and Interactional Dimensions of Human–Computer Interaction*, pp. 67–106. New York: Cambridge University Press.

ANDERSON, J. R. (1990). *The Adaptive Character of Thought*. Hillsdale, NJ: Erlbaum.

BENNETT, J. B. (1984). Managing to meet usability requirements: establishing and meeting software development goals. In J. BENNETT, D. CASE, J. SANDELIN & M. SMITH, Eds. *Visual Display Terminals*, pp. 161–184. Englewood Cliffs, NJ: Prentice-Hall.

BENTLEY, R., HUGHES, J. A., RANDALL, D., RODDEN, T., SAWYER, P., SHAPIRO, D. & SOMMERVILLE, I. (1992). Ethnographically-informed system design for air traffic control. *Proceedings of CSCW'92: Computer Supported Cooperative Work*, Toronto, pp. 123–129. New York: Association for Computing Machinery.

BIAS, R. G. & MAYHEW, D. J., Eds. (1994). *Cost-Justifying Usability*. Boston, MA: Academic Press.

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. In A. F. MONK & N. GILBERT, Eds. *Perspectives on HCI: Diverse Approaches*, pp. 223–248. London: Academic Press.

BLOMBERG, J. L. (1995). Ethnography: aligning field studies of work and system design. In A. F. MONK & N. GILBERT, Eds. *Perspectives on HCI: Diverse Approaches*, pp. 175–197. London: Academic Press.

BLY, S., HARRISON, S. & IRWIN, S. (1993). Media spaces: bringing people together in a video, audio, and computing environment. *Communications of the ACM*, **36,** 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. In G. BJERKNES, P. EHN & M. KYNG, Eds. *Computers and Democracy: A Scandinavian Challenge*, pp. 251–278. Brookfield, VT: Avebury.

BOWKER, G., STAR, S. L., GASSER, L. & TURNER, B., Eds. (1995). *Social Science Research, Technical Systems and Cooperative Work*. Mahwah, NJ: Erlbaum.

BROOKS, F. P. (1975/1995). *The Mythical Man-Month: Essays on Software Engineering* (1995 Anniversary Edn). Reading, MA: Addison-Wesley.

BROOKS, R. (1993). The case for the specialized interface. *IEEE Software*, **10,** 86–88.

BRUNSWICK, E. (1956). *Perception and the Representative Design of Psychological Experiments*. Berkeley, CA: University of California Press.

BUCKINGHAM SHUM, S. (1996). Analyzing the usability of a design rationale notation. In T. P. MORAN & J. M. CARROLL, Eds. *Design Rationale: Concepts, Techniques, and Use*, pp. 185–215. Mahwah, NJ: Erlbaum.

BUTLER, K. A. (1985). Connecting theory and practice: a case study of achieving usability goals. *Proceedings of CHI '85 Human Factors in Computing Systems,* San Francisco, pp. 85–88. New York: Association for Computing Machinery.

CARD, S. K., MORAN, T. P. & NEWELL, A. (1983). *The Psychology of Human–Computer Interaction*. Hillsdale, NJ: Erlbaum.

CARROLL, J. M. (1985). *What's in a Name? An Essay in the Psychology of Reference*. New York: Freeman.

CARROLL, J. M. (1989). Evaluation, description and invention: paradigms for human–computer interaction. In M. C. YOVITS, Ed. *Advances in Computers*, Vol. 29, pp. 47–77. San Diego, CA: Academic Press.

CARROLL, J. M. (1990). *The Nurnberg Funnel: Designing Minimalist Instruction for Practical Computer Skill*. Cambridge, MA: MIT Press.

CARROLL, J. M., Ed. (1991). *Designing Interaction: Psychology at the Human–Computer Interface*. New York: Cambridge University Press.

CARROLL, J. M., Ed. (1995). *Scenario-based Design: Envisioning Work and Technology in System Development.* New York: Wiley.

CARROLL, J. M., ALPERT, S. R., KARAT, J., VAN DEUSEN, M. & ROSSON, M. B. (1994). Capturing design history and rationale in multimedia narratives. In *Proceedings of CHI'94: Human Factors in Computing Systems*, Boston, pp. 192–197. New York: Association for Computing Machinery/Addison-Wesley.

CARROLL, J. M. & CAMPBELL, R. L. (1986). Softening up hard science: reply to Newell and Card. *Human–Computer Interaction*, **2,** 227–249.

CARROLL, J. M. & CARRITHERS, C. (1984). Blocking learner errors in a training wheels system. *Human Factors*, **26,** 377–389.

CARROLL, J. M., MACK, R. L. & KELLOGG, W. A. (1988). Interface metaphors and user interface design. In M. HELANDER, Ed. *Handbook of Human–Computer Interaction*, pp. 67–85. Amsterdam: North-Holland.

CARROLL, J. M. & ROSSON, M. B. (1985). Usability specifications as a tool in iterative development. In H. R. HARTSON, Ed. *Advances in Human–Computer Interaction*, Vol. 1, pp. 1–28. Norwood, NJ: Ablex.

CARROLL, J. M. & ROSSON, M. B. (1991). Deliberated evolution: stalking the view matcher in design space. *Human–Computer Interaction*, **6,** 281–318.

CARROLL, J. M. & ROSSON, M. B. (1992). Getting around the task-artifact cycle: how to make claims and design by scenario. *ACM Transactions on Information Systems*, **10,** 181–212.

CARROLL, J. M. & ROSSON, M. B. (1995). Managing evaluation goals for training. *Communications of the ACM,* **38,** 40–48.

CARROLL, J. M., SINGLEY, M. K. & ROSSON, M. B. (1992). Integrating theory development with design evaluation. *Behaviour and Information Technology*, **11,** 247–255.

CARROLL, J. M. & THOMAS, J. C. (1982). Metaphor and the cognitive representation of computing systems. *IEEE Transactions on Systems, Man and Cybernetics*, **12,** 107–116.

CARROLL, J. M., THOMAS, J. C. & MALHOTRA, A. (1979). A clinical-experimental analysis of design problem solving. *Design Studies*, **1,** 84–92.

CONKLIN, J. & BEGEMAN, M. (1988). gIBIS: a hypertext tool for exploratory policy discussion. *ACM Transactions on Office Information Systems*, **6,** 303–331.

CONKLIN, J. & YAKEMOVIC, K. C. B. (1991). A process-oriented approach to design rationale. *Human–Computer Interaction*, **6,** 357–391.

CURTIS, B., KRASNER, H. & ISCOE, N. (1988). A field study of the software design process for large systems. *Communications of the ACM*, **31,** 1268–1287.

CYPHER, A., HALBERT, D., KURLANDER, D., LIEBERMAN, H., MAULSBY, D., MYERS, B. & TURRANSKY, A., Eds. (1993). *Watch What I do: Programming by Demonstration*. Cambridge, MA: MIT Press.

DE GROOT, A. D. (1965). *Thought and Choice in Chess*. Paris: Mouton.

DENNING, P. J., COMER, D. E., GRIES, D., M ULDER, M. C., TUCKER, A. B., TURNER, A. J. & YOUNG, P. R. (1989). Computing as a discipline. *Communications of the ACM*, **32,** 9–23.

DIX, A., FINLAY, J., ABOWD, G. & BEALE, R. (1993). *Human–Computer Interaction*. Englewood Cliffs, NJ: Prentice-Hall.

DOUGLAS, S. A. & MORAN, T. P. (1983). Learning text editor semantics by analogy. *Proceedings of CHI'83 Conference on Human Factors in Computing Systems,* Boston, pp. 207–211. New York: Association for Computing Machinery.

DREYFUSS, H. (1955). *Designing for People*. New York: Simon and Schuster.
EHN, P. (1988). *Work-oriented Design of Computer Artifacts*. Stockholm: Arbetslivscentrum.
ENGESTROM, Y. (1993). Developmental work research: reconstructing expertise through expansive learning. In M. NURMINEN & G. WEIR, Eds. *Human Jobs and Computer Interfaces*. Amsterdam: Elsevier.
ESPER, E. A. (1925). A technique for the experimental investigation of associative interference in artificial linguistic material. *Language Monographs*, **1,** 1–47.
ERICSSON, K. A. & SIMON, H. A. (1985). *Protocol Analysis: Verbal Reports as Data*. Cambridge, MA: The MIT Press.
FLANAGAN, J. C. (1954). The critical incident technique. *Psychological Bulletin*, **51,** 28–35.
FISCHER, G., LEMKE, A. C., MCCALL, R. & MORCH, A. I. (1991). Making argumentation serve design. *Human–Computer Interaction*, **6,** 393–419.
FURNAS, G., LANDAUER, T. K., GOMEZ, L. & DUMAIS, S. (1983). Statistical semantics: Analysis of the potential performance of keyword information systems. *Bell System Technical Journal*, **62,** 1753–1806.
GALLOPOULOS, E., HOUSTIS, E. & RICE, J. R. (1994). Computer as thinker/doer: problem-solving environments for computational science. *IEEE Computational Science & Engineering*, **1,** 11–23.
GASSER, L. (1986). The integration of computing and routine work. *ACM Transactions on Office Information Systems*, **4,** 205–225.
GIBSON, J. J. (1979). *The Ecological Approach to Visual Perception*. Boston, MA: Houghton Mifflin.
GOOD, M., SPINE, T. M., WHITESIDE, J. & GEORGE, P. (1986). User-derived impact analysis as a tool for usability engineering. *Proceedings of CHI'86 Conference on Human Factors in Computing Systems,* Boston, pp. 241–246. New York: Association for Computing Machinery.
GOULD, J. D. & BOIES, S. J. (1983). Human factors challenges in creating a principal support office system—the speech filing approach. *ACM Transactions on Office Information Systems*, **1,** 273–298.
GRAY, W. D., JOHN, B. E. & ATWOOD, M. E. (1992). The precis of project Ernestine, or, An overview of a validation of GOMS. *Proceedings of CHI'92 Conference on Human Factors in Computing Systems,* Monterey, CA, pp. 307–312, New York: Association for Computing Machinery.
GREATBATCH, D., HEATH, C., LUFF, P. & CAMPION, P. (1995). Conversation analysis: Human–computer interaction and the general practice consultation. In A. F. MONK & N. GILBERT, Eds. *Perspectives on HCI: Diverse Approaches*, pp. 199–222. London: Academic Press.
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. *Communications of the ACM,* **32,** 1164–1173.
GRUDIN, J. (1994). Groupware and social dynamics: eight challenges for developers. *Communications of the ACM*, **37,** 92–105.
HILTZ, S. R. & TUROFF, M. (1978/1993). *The Network Nation: Human Communication via Computer*. (1993 Revised Edn). Cambridge, MA: MIT Press.
HUGHES, J., KING, V., RODDEN, T. & ANDERSON, H. (1994). Moving out from the control room: ethnography in system design. *Proceedings of CSCW'94 Conference on Computer-Supported Cooperative Work*, Chapel Hill, NC, pp. 429–439, New York: Association for Computing Machinery.
HUTCHINS, E., HOLLAN, J. & NORMAN, D. A. (1986). Direct manipulation interfaces. In D. A. NORMAN & S. W. DRAPER, Eds. *User Center System Design*, pp. 87–124. Hillsdale, NJ: Erlbaum.
JEFFRIES, R. J., MILLER, J. R., WHARTON, C. & UYEDA, K. M. (1991). User interface evaluation in the real world: a comparison of four techniques. *Proceedings of CHI'91 Human Factors in Computing Systems*, New Orleans, pp. 119–124. New York: Association for Computing Machinery.
JOHN, B. E. (1990). Extensions of GOMS analyses to expert performance requiring perception of dynamic visual and auditory information. *Proceedings of CHI'90 Human Factors in Computing Systems,* Seattle, pp. 107–115. New York: Association for Computing Machinery.

KARAT, C. M., CAMPBELL, R. L. & FIEGEL, T. (1992). Comparison of empirical testing and walkthrough methods in user interface evaluation. *Proceedings of CHI'92 Human Factors in Computing Systems*, Monterey, CA, pp. 397–404. New York: Association for Computing Machinery.

KIERAS, D. E. (1988). Towards a practical GOMS model methodology for user interface design. In M. HELANDER, Ed. *Handbook of Human–Computer Interaction*, pp. 135–157. Amsterdam: North-Holland.

KLING, R. (1991). Cooperation, coordination and control in computer-supported work. *Communications of the ACM*, **34,** 83–88.

KLING, R. & JEWETT, T. (1994). The social design of worklife with computers and networks: A natural systems perspective. In M. YOVITZ, Ed. *Advances in computers*, Vol. 39. Orlando, FL: Academic Press.

KNORR-CETINA, K. D. (1981). The micro-sociological challenge of macro-sociology: towards a reconstruction of social theory and methodology. In K. D. KNORR-CETINA & A. CICOUREL, Eds. *Advances in Social Theory and Methodology: Towards an Integration of Micro- and Macro-Sociologies*. London: Routledge and Kegan Paul.

KUHN, S. & MULLER, M. J., Eds. (1993). Special section on participatory design. *Communications of the ACM,* **36,** 24–103.

KUUTTI, K. & ARVONEN, T. (1992). Identifying CSCW applications by means of activity theory concepts: a case example. *Proceedings of CSCW'92 Computer Supported Cooperative Work*, Toronto, pp. 233–240. New York: Association for Computing Machinery.

LAVE, J. (1988). *Cognition in Practice: Mind, Mathematics, and Culture*. New York: Cambridge University Press.

LEIGH, J., JOHNSON, A. E., VASILAKIS, C. A. & DEFANTI, T. A. (to appear). Multi-perspective collaborative design in persistent networked virtual environments. *Proceedings of IEEE Virtual Reality Annual International Symposium, VRAIS 96*, Santa Clara.

LEWIS, C. H. & NORMAN, D. A. (1986). Designing for error. In D. A. NORMAN & S. W. DRAPER, Eds. *User Center System Design*, pp. 411–432. Hillsdale, NJ: Erlbaum.

MACK, R. L., LEWIS, C. H. & CARROLL, J. M. (1983). Learning to use office systems: problems and prospects. *ACM Transactions on Office Information Systems*, **1,** 254–271.

MACLEAN, A., YOUNG, R. M., BELLOTTI, V. M. E. & MORAN, T. P. (1991). Questions, options, and criteria: elements of design space analysis. *Human–Computer Interaction,* **6,** 201–250

MALHOTRA, A., THOMAS, J. C., CARROLL, J. M. & MILLER, L. A. (1980). Cognitive processes in design. *International Journal of Man–Machine Interaction*, **12,** 119–140.

MILLER, L. A. (1974). Programming by non-programmers. *International Journal of Man–Machines Studies*, **6,** 237–260.

MONK, A. M., CARROLL, J. M., HARRISON, M., LONG, J. & YOUNG, R. M. (1990). New approaches to theory in HCI: how should we judge their acceptability? *Proceedings of INTERACT'90 Human–Computer Interaction,* Cambridge, pp. 1055–1058. Amsterdam: North-Holland.

MONK, A. M., NARDI, B., GILBERT, N., MANTEI, M. & MCCARTHY, J. (1993). Mixing oil and water? Ethnography versus experimental psychology in the study of computer-mediated communication. *Proceedings of InterCHI'93 Human Factors in Computing Systems*, Amsterdam, pp. 3–6. New York: Association for Computing Machinery.

MORAN, T. P. (1983). Getting into a system: external–internal task mapping analysis. *Proceedings of CHI'83 Human Factors in Computing Systems*, Boston, pp. 45–49. New York: Association for Computing Machinery.

MORAN, T. P. & CARROLL, J. M., Eds. (1996). *Design Rationale: Concepts, Techniques, and Use*. Mahwah, NJ: Erlbaum.

NARDI, B. A., Ed. (1995*a*). *Context and Consciousness: Activity Theory and Human–Computer Interaction*. Cambridge, MA: MIT Press.

NARDI, B. A. (1995*b*). Studying context: A comparison of activity theory, situated action models and distributed cognition. In B. A. NARDI, Ed. *Context and Consciousness: Activity Theory and Human–Computer Interaction*, pp. 69–102. Cambridge, MA: MIT Press.

NEWELL, A. & CARD, S. K. (1985). The prospects for psychological science in human–computer interaction. *Human–Computer Interaction*, **1,** 209–242.

NEWELL, A. & SIMON, H. A. (1972). *Human Problem Solving*. Englewood Cliffs, NJ: Prentice-Hall.
NIELSEN, J. Ed. (1989). *Coordinating User Interfaces for Consistency*. New York: Academic Press.
NIELSEN, J. (1993). *Usability Engineering*. Boston, MA: Academic Press.
NIELSEN, J. (1994). Estimating the number of subjects needed for a thinking aloud test. *International Journal of Human–Computer Studies*, **41,** 385–397.
NIELSEN, J. & MACK, R. L., Eds. (1994). *Usability Inspection Methods*. New York: Wiley.
NIELSEN, J. & MOLLICH, R. (1990). Heuristic evaluation of user interfaces. *Proceedings of CHI'90 Conference on Human Factors in Computing Systems*, Seattle, pp. 249–256. New York: Association for Computing Machinery.
NIELSEN, J. & PHILLIPS, V. L. (1993). Estimating the relative usability of two interfaces: heuristic, formal and empirical methods compared. *Proceedings of INTERCHI'93 Conference on Human Factors in Computing Systems*, Amsterdam, pp. 214–221. New York: Association for Computing Machinery.
NORMAN, D. A. (1991). Cognitive artifacts. In J. M. CARROLL, Ed. *Designing Interaction*: *Psychology at the Human–Computer Interface*, pp. 17–38. New York: Cambridge University Press.
NORMAN, D. A. & DRAPER, S. W., 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. *Communications of the ACM*, **34,** 40–61.
NYGAARD, K. (1979). The iron and metal project: trade union participation. In A. SANDBERG, Ed. *Computers Dividing Man and Work: Recent Scandinavian Research on Planning and Computers from a Trade Union Perspective*, p. 98. Malmö: Arbetslivscentrum.
PAVA, C. H. P. (1983). *Managing New Office Technology: an Organizational Strategy*. New York: Free Press.
PAYNE, S. J. & GREEN, T. R. G. (1989). Task-action grammars: the model and its developments. In D. DIAPER, Ed. *Task Analysis for Human–Computer Interaction*, pp. 75–107. Chichester: Ellis Horwood.
PAYNE, S. J., SQUIBB, H. & HOWES, A. (1990). The nature of device models: the yoked state space hypothesis and some experiments with text editors. *Human–Computer Interaction,* **5,** 415–444.
POLSON, P. G., LEWIS, C. H., RIEMAN, J. & WHARTON, C. (1992). Cognitive walkthroughs: A method for theory-based evaluation of user interfaces. *International Journal of Man–Machine Studies*, **36,** 741–773.
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 Science*, **4,** 155–169.
ROSSON, M. B. & CARROLL, J. M. (1995). Narrowing the specification-implementation gap in scenario-based design. In J. M. CARROLL, Ed. *Scenario-based Design*: *Envisioning Work and Technology in System Development*, pp. 247–278. New York: Wiley.
ROYCE, W. W. (1970). Managing the development of large software systems: concepts and techniques. *Proceedings of Western Electric Show and Convention*, Westcon, Los Angeles, pp. (A/1) 1-(A/1) 9 (Reprinted in: *Proceedings of the 11th International Conference on Software Engineering*, Pittsburgh, May, pp. 328–338.
SCRIVEN, M. (1967). The methodology of evaluation. In R. TYLER, R. GAGNE & M. SCRIVEN, Eds. *Perspectives of Curriculum Evaluation*, pp. 39–83. Chicago, IL: Rand McNally.
SCHULER, D. (1996). *New Community Networks: Wired for Change*, IL. New York: Addison-Wesley, Reading.
SCHULER, D. & NAMIOKA, A., Eds. (1993). *Participatory Design: Principles and Practices*. Hillsdale, NJ: Erlbaum.
SHATTUCK, L. & WOODS, D. (1994). The critical incident technique: 40 years later. *Proceedings of the Human Factors and Ergonomics Society 38th Annual Meeting*, Nashville, pp. 1080–1084. Santa Monica, CA: Human Factors and Ergonomics Society.
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,** 57–62.

SHNEIDERMAN, B., MAYER, R., MCKAY, D. & HELLER, P. (1977). Experimental investigations of the utility of detailed flowcharts in programming. *Communications of the ACM*, **20,** 373–381.

SIME, M. E., GREEN, T. G. R. & GUEST, D. J. (1973). Psychological evaluation of two conditional constructions used in computer languages. *International Journal of Man–Machine Studies,* **5,** 105–113.

SIMON, H. A. (1969). *The Sciences of the Artificial*. Cambridge, MA: MIT Press.

SMITH, D. C., IRBY, C., KIMBALL, R., VERPLANK, B. & HARSLEM, E. (1982). Designing the Star user interface. *Byte*, **7,** 242–282.

SOLOWAY, E., GUZDIAL, M. & HAY, K. E. (1994). Learner-centered design: the challenge for HCI in the 21st century. *Interactions*, **1,** 36–48.

SPROULL, L. & KIESLAR, S. (1991). *Connections: New Ways of Working in a Networked Organization*. Cambridge, MA: MIT Press.

SUCHMAN, L. A. (1987). *Plans and Situated Actions: The Problem of Human–Machine Communication*. New York: Cambridge University Press.

SUCHMAN, L. A., Ed. (1995). Special section on representations of work. *Communications of the ACM*, **38,** 33–68.

SUKAVIRIYA, P., FOLEY, J. D. & GRIFFITH, T. (1993). A second-generation user interface design environment: the model and runtime architecture. *Proceedings of INTERCHI'93 Conference on Human Factors in Computing Systems*, Amsterdam, pp. 375–382. New York: Association for Computing Machinery.

SUTCLIFFE, A., CARROLL, J. M., YOUNG, R. M. & LONG, J. (1991). HCI theory on trial. *Proceedings of INTERCHI'93 Conference on Human Factors in Computing Systems*, Amsterdam, pp. 375–382. New York: Association for Computing Machinery.

SZEKELY, P., LUO, P. & NECHES, R. (1993). Beyond interface builders: model-based interface tools. *Proceedings of CHI'91 Conference on Human Factors in Computing Systems*, New Orleans, pp. 399–401. New York: Association for Computing Machinery.

TANNER, P. P. & BUXTON, W. A. S. (1985). Some issues in future user interface management system (UIMS) development. In G. E. PFAFF, Ed. *User Interface Management Systems: Proceedings of the Workshop on User Interface Management Systems*, Seehiem, FRG, November 1–3, pp. 67–79. New York: Springer.

THOMAS, P. J., Ed. (1995). *The Social and Interactional Dimensions of Human–Computer Interaction*. New York: Cambridge University Press.

TUCKER, A. B., Ed. (1997). *The Handbook of Computer Science and Engineering*. Boca Raton, FL: CRC Press.

TUCKER, A. B. & TURNER, A. J. (1991). A summary of the ACM/IEEE-CS joint curriculum task force report: computing curricula (1991). *Communications of the ACM*, **34,** 68–84.

VIRZI, R. A. (1992). Refining the test phase of usability evaluation: how many subjects is enough? *Human Factors*, **34,** 457–468.

WASSERMAN, A. I. & SHEWMAKE, D. T. (1982). Rapid prototyping of interactive information systems. *ACM Software Engineering Notes*, **7,** 171–180.

WASSERMAN, A. I. & SHEWMAKE, D. T. (1985). The role of prototypes in the user software engineering (USE) methodology. In H. R. HARTSON, Ed. *Advances in Human–Computer Interaction*, Vol. 1, pp. 191–209. Norwood, NJ: Ablex.

WEISSMAN, L. (1974). *A methodology for Studying the Psychological Complexity of Computer Programs*. Ph.D. Thesis, University of Toronto, Toronto, Canada.

WERTSCH, J. (1985). *Vygotsky and the Social Formation of Mind*. Cambridge, MA: Harvard University Press.

WHITESIDE, J., BENNETT, J. & HOLTZBLATT, K. (1988). Usability engineering: our experience and evolution. In M. HELANDER, Ed. *Handbook of Human–Computer Interaction*, pp. 791–817. Amsterdam: North-Holland.

WHITESIDE, J. & 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,* pp. 337–352. Cambridge, MA: Bradford/MIT Press.

WILLIGES, R. C., WILLIGES, B. H. & HAN, S. H. (1993). Sequential experimentation in human–computer interface design. In H. R. HARTSON & D. HIX, Eds. *Advances in Human–Computer Interaction*, Vol. 4, pp. 1–30. Norwood, NJ: Ablex.

WIXON, D., HOLTZBLATT, K. & KNOX, S. (1990). Contextual design: an emergent view of system design. In *Proceedings of CHI'90: Human Factors in Computing Systems*, Seattle, pp. 329–336. New York: Association for Computing Machinery.

WRIGHT, R. B. & CONVERSE, S. A. (1992). Method bias and concurrent verbal protocol in software usability testing. *Proceedings of the Human Factors and Ergonomics Society 36th Annual Meeting*, Atlanta, pp. 891–912. Santa Monica, CA: Human Factors and Ergonomics Society.

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

Paper accepted for publication by Editor, B. R. Gaines.