User Modeling and User-Adapted Interaction **11:** 65–86, 2001.
© 2001 *Kluwer Academic Publishers. Printed in the Netherlands.*

User Modeling in Human–Computer Interaction

GERHARD FISCHER
Center for LifeLong Learning & Design (L^3D), Department of Computer Science and Institute of Cognitive Science, University of Colorado at Boulder, Boulder, CO 80309, U.S.A.
E-mail: gerhard@cs.colorado.edu

(Received: 17 January 2000; in revised form: 1 May 2000)

Abstract. A fundamental objective of human–computer interaction research is to make systems more usable, more useful, and to provide users with experiences fitting their specific background knowledge and objectives. The challenge in an information-rich world is not only to make information available to people at any time, at any place, and in any form, but specifically to say the "right" thing at the "right" time in the "right" way. Designers of collaborative human–computer systems face the formidable task of writing software for millions of users (at design time) while making it work as if it were designed for each individual user (only known at use time).

User modeling research has attempted to address these issues. In this article, I will first review the objectives, progress, and unfulfilled hopes that have occurred over the last ten years, and illustrate them with some interesting computational environments and their underlying conceptual frameworks. A special emphasis is given to high-functionality applications and the impact of user modeling to make them more usable, useful, and learnable. Finally, an assessment of the current state of the art followed by some future challenges is given.

Key words: user modeling, human computer interaction, collaborative human–computer systems, high functionality applications, adaptive and adaptable systems, active help systems, critiquing systems, design environments

1. Introduction

User modeling is one of a number of research areas that intuitively seem to be winning propositions and worthwhile investments based on their obvious need and potential payoff. Other domains comparable to user modeling are (1) software reuse (e.g. reuse can be justified by the fact that complex systems develop faster if they can build on stable subsystems); and (2) organizational memories and organizational learning (e.g. creating socio-technical environments that help to transcend the limitation of the individual human mind). These approaches seem to be appealing, natural, theoretically justifiable, desirable, and needed. But in reality, progress in these areas has been slow and difficult, and success stories are rare.

In this article I first analyze the evolution of human–computer interaction (HCI) as a research area over the last 15 years and briefly characterize problems for HCI for which user modeling may provide some answers. I describe a set of selected themes and examples illustrating user modeling approaches in HCI. I conclude by giving a brief assessment of the present state of user modeling and by enumerating a few challenges for the future. This paper is closely related to two other contributions in this volume: "Learner Control" by Judy Kay (Kay, 2001) and "Adaptive Techniques for Universal Access" by Constantine Stephanidis (Stephanidis, 2001).

2. The Evolution of Human–Computer Interaction

HCI studies the interactions and the relationships between humans and computers. HCI is more than user interfaces and more than "screen-deep" (Computer Science and Technology Board – National Research Council, 1997); it is a multidisciplinary field covering many areas (Helander et al., 1997). In the first ten to fifteen years of its history, HCI has focused on interfaces (particularly on the possibilities and design criteria for graphical user interfaces (GUIs) using windows, icons, menus, and pointing devices (WIMPs)) to create more usable systems. As interface problems were better understood, the primary HCI concerns started to shift beyond the interface (to respond to observations as articulated by D. Engelbart: "*If ease of use was the only valid criterion, people would stick to tricycles and never try bicycles*"). More recent HCI research objectives (Fischer, 1993a) are concerned with tasks, with shared understanding, and with explanations, justifications, and argumentation about actions, and not just with interfaces. The new essential challenges are improving the way people use computers to work, think, communicate, learn, critique, explain, argue, debate, observe, decide, calculate, simulate, and design.

2.1. Collaborative human–computer systems

Some of the beginnings of user modeling were derived from the need and desire to provide better support for human–computer collaboration. Collaboration in this context is defined as "a process in which two or more agents work together to achieve shared goals" (Terveen, 1995). Some fundamental issues (such as shared goals, shared context, control, (co)-adaptation, (co)-evolution, and learning) can be derived from this definition. Human–computer collaboration can be approached from two different perspectives: an emulation approach and a complementing approach. The *emulation approach* is based on the metaphor that to improve human–computer collaboration is to endow computers with "human-like abilities". The *complementing approach* is based on the fact that computers are not human and that human-centered design should exploit the asymmetry of human and computer by developing new interaction and collaboration possibilities (Suchman, 1987).

Historically, the major emphasis in user modeling has focused on the human emulation approach (see, for example, Kobsa and Wahlster, 1989, and the section "user and discourse modeling" in Maybury and Wahlster, 1998). However, based on the limited success of the emulating approach, the interest has shifted more and more to the complementing approach (Bobrow, 1991). There is growing evidence that the problems of user modeling in the complementing approach are more

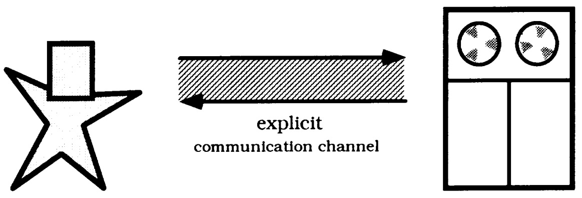


Figure 1. The human–computer dyad

tractable, more feasible, and more desirable, as evidenced by their increasing influence in the design of commercial high-functionality applications (Horvitz et al., 1998). A similar shift taking place in the Intelligent Tutoring Systems (ITS) community is described in the contribution by Kay (Kay, 2001).

2.2. From novice to skilled domain worker

The original HCI approaches, by being focused on making systems more *usable*, have often reduced the expressive power of the systems and of interfaces to accommodate novices and casual users who are assumed to be using the system for the first time, for only a few times, and for simple activities. Walk-up-and-use systems, such as ATMs (Automated Teller Machines), are examples of low-threshold, low-ceiling systems; they should be easy to understand and use without prior experience. Complex systems for professional use need to be *useful*; they must allow their users to do the tasks they have to do to get their jobs done. These professional worlds are complex, leading to *high-functionality applications* (HFA). These systems are often difficult to use at first, but over time users are able to perform a wide range of tasks with the system. Generic assumptions about users may be adequate in systems for novices (the design criteria being based on generic cognitive functions, as, for example, defined by the Model Human Processors (Card et al., 1983)), but only if we ignore the requirements to provide universal access for people with different (dis)abilities (Stephanidis, 2001). Generic assumptions about skilled domain workers being the primary users of HFAs are definitely limiting the learnability and usability of these systems.

2.3. Knowledge-based HCI

Traditionally, computer usage was modeled as a human–computer dyad in which the two were connected by a narrow *explicit* communication channel (see Figure 1), such as text-based terminals in a time-sharing environment.

The advent of more sophisticated interface techniques, such as windows, menus, pointing devices, color, sound, and touch-screens have widened this explicit communication channel. In addition to exploring the possibilities of new design possibilities for the explicit communication channel, knowledge-based architectures

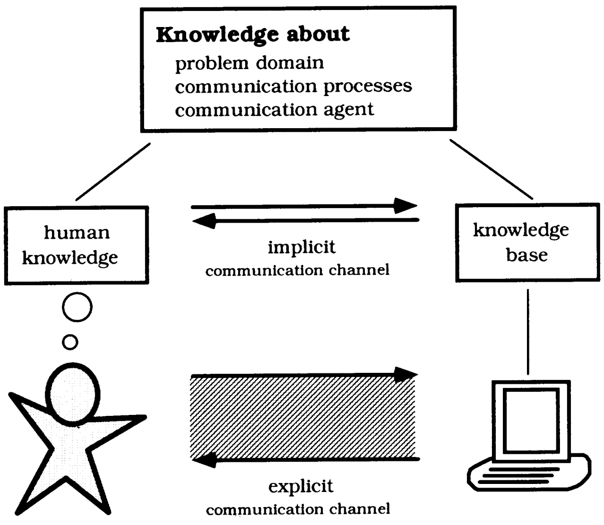


Figure 2. Knowledge-based HCI

for HCI have explored the possibility of an *implicit* communication channel (see Figure 2).

The implicit communication channel supports communication processes that require the computer to be provided with a considerable body of knowledge about problem domains, about communication processes, and about the agents involved.

- *Knowledge about the problem domain:* Shared knowledge builds upon large amounts of knowledge about specific domains. This knowledge constrains the number of possible actions and describes reasonable goals and operations in the domain of specific users, thereby supporting human problem-domain interaction and not just human–computer interaction (Fischer, 1994; Horvitz et al., 1998).
- *Knowledge about communication processes:* The information structures that control communication should be accessible and changeable by the user. A knowledge-based HCI system should have knowledge about when and whether to assist the user, interrupt the user, and volunteer information to the user contextualized to the task at hand (Fischer and Stevens, 1987; Horvitz, 1999).
- *Knowledge about the communication agent:* The "typical" user of a system does not exist; there are many different kinds of users, and the requirements of an individual user usually change with experience (Mackay, 1991). Simple classification schemes based on stereotypes (Rich, 1989), such as novice, intermediate, or expert users, are inadequate for complex knowledge-based systems because these attributes become dependent on a particular context rather than applying to users globally. One of the central objectives of user modeling

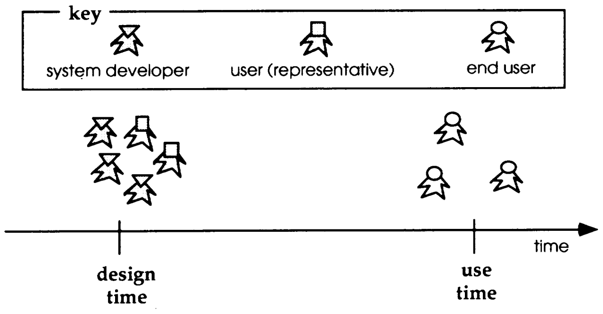


Figure 3. Design and use time

in HCI is to address the problem that systems will be unable to interact with users cooperatively unless they have some means of finding out what the user really knows and does. Techniques to achieve this include: (1) being told by the users (e.g. by questionnaires, setting preferences, or specification components (Nakakoji, 1993)); (2) being able to infer it from the user's actions (e.g. by using critics (Fischer et al., 1991; Mastaglio, 1990)) or usage data (Adachi, 1998; Hill et al., 1992); and (3) communicating information about external events to the system (Bolt, 1984; Harper et al., 1992).

2.3. Design time and use time

One of the fundamental problems of system design is: how do we write software for millions of users (at design time), while making it work as if it were designed for each individual user (who is known only at use time)? Figure 3 differentiates between two stages in the design and use of a system. At design time, developers create systems, and they have to make decisions for users for situational contexts and for tasks that they can only anticipate. For print media, a fixed context is decided at design time whereas for computational media, the behavior of a system at use time can take advantage of contextual factors (such as the background knowledge of a user, the specific goals and objectives of a user, the work context, etc.) *only known at use time*. The fundamental difference is that computational media have interpretive power: they can analyze the artifacts created by users and the interaction patterns between users and system, and they can support users in their articulation of additional contextual factors.

An important point about user modeling might be that use time and design time get blurred. If the system is constantly adapting or is being adapted to users, use time becomes a different kind of design time (Henderson and Kyng, 1991). The need to support a broad class of different users leads to high-functionality applications with all their associated possibilities and problems. A feasible design strategy to support users in their *own domain of knowledge* is that system designers make assumptions about classes of users and sets of tasks in which they want to engage – a design methodology leading to domain-oriented systems (Fischer, 1994).

2.4. Saying the "right" thing at the "right" time in the "right" way

The challenge in an information-rich world (in which human attention is the most valuable and scarcest commodity*)* is not only to make information available to people at any time, at any place and in any form, but to reduce information overload by making information relevant to the task-at-hand and to the assumed background knowledge of the users. Techniques to say *the "right" thing* include: (1) support for differential descriptions that relate new information to information and concepts assumed to be known by a specific user; and (2) embedded critiquing systems (Fischer et al., 1998) that make information more relevant to the task-at-hand by generating interactions in real time, on demand, and suited to individual users as needed. They are able to do so by exploiting a richer context provided by the domain orientation of the environment, by the analysis of partially constructed artifacts and partially completed specifications. To say things at *the "right" time* requires to balance the costs of intrusive interruptions against the loss of context-sensitivity of deferred alerts (Horvitz et al., 1999). To say things in *the "right" way* (for examples by using multimedia channel to exploit different sensory channels) is especially critical for users who may suffer from some disability (Stephanidis, 2001).

3. User Modeling in HCI

Some HCI researchers have been interested in user modeling because there is the potential that user modeling techniques will improve the collaborative nature of human–computer systems. In the context of this article, *user models* are defined as models that systems have of users that reside inside a computational environment. They should be differentiated from mental models that users have of systems and tasks that reside in the heads of users, in interactions with others and with artifacts (the models D_1, D_2, and D_3 in Figure 4 are examples of mental models). Descriptions of some specific user modeling attempts and challenges in HCI follow.

3.1. How the west was won – an early success example of user modeling

The WEST system (Burton and Brown, 1982) represents an early pioneering effort to explore issues associated with user modeling. WEST was a coaching system for a game called "How the West was Won" that was modeled on "Chutes and Ladders." The players rotate three spinners and have to form an arithmetic expression from the three numbers that turn up on the spinners using $+$, $-$, $*$, / and appropriate parentheses (and they have to specify what the value of the expression is). So, for example, a player who gets a 2, 3, and 4 on the spinners, might form the expression

$(2 + 3)^*4 = 20$ and move forward 20 spaces. If players land on a town (towns occur every 10 spaces), they move forward to the next town. If they land on a chute, they slide to the end of the chute. If they land on an opponent, that opponent is sent back two towns. The optimal strategy is to figure out all the possible moves and take the one that puts you farthest ahead of your opponents. But empirical analyses showed that students did not use this strategy; they were much more likely to rely on a strategy such as adding the two smallest numbers and multiplying by the largest.

The WEST coach analyzed students' moves in terms of the optimal strategy and could rate the moves with respect to that strategy. It watched to see if the students consistently followed a less-than-optimal strategy, such as not taking opportunities to land on a town or chute or opponent. If the WEST coach detected such a pattern, it would intervene at an opportune time, when the student's move was far from optimal, and it would point out how the student could have done much better. It then would give the student a chance to take the move over.

In the context of WEST, the following problems of user modeling were explored:

- *shared context*: computer coaches are restricted to inferring the students' shortcomings from whatever they do in the context of playing the game or solving the problem;
- *initiative and intrusiveness*: the user model was used (1) to make a judgment of when to give valuable advice and make relevant comments to students without being so intrusive as to destroy the fun of the game; and (2) to avoid the danger that students will never develop the necessary skills for examining their own behavior and looking for the causes of their own mistakes because the coach immediately points out the students' errors;
- *relevance*: WEST developed the paradigm "coaching by issues and examples." By assessing the situational context and acting accordingly, students were coached in a way in which they could see the usefulness of the issue at a time when they were most receptive to the idea being presented. Based on the information contained in the user model, the system used explicit intervention and tutoring strategies to enable the system to say the "right" thing at the "right" time.

Although the WEST system explored some of the basic concepts of user modeling, it did so in the context of a very simple domain in which outcomes were limited to the combinatorics of a few variables. The approach worked well because the computer expert (as one component of the overall system) operating in a "closed-world" can play an optimal game, and it can determine the complete range of alternative behaviors. Low level, individual events were easy to interpret. The user model was incrementally constructed by exploiting many events occurring in the same domain.

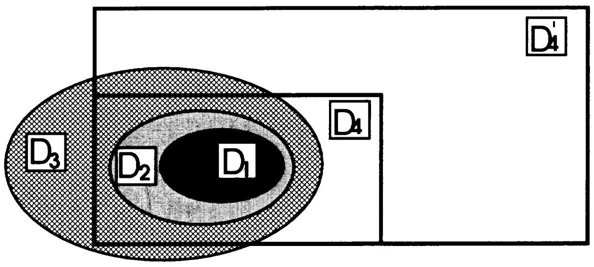


Figure 4. Levels of users' knowledge about a high-functionality application

3.2. HIGH-FUNCTIONALITY APPLICATIONS

High-functionality applications (such as Unix, MS-Office, Photoshop, Eudora, etc.) are used to model parts of existing worlds and to create new worlds. They are complex systems because they serve the needs of large and diverse user populations. If you asked 100 different people what features they would like to have in a particular application, you would end up with a very large number of features. The design of HFAs must address three problems: (1) the unused functionality must not get in the way; (2) unknown existing functionality must be accessible or delivered at times when it is needed; and (3) commonly used functionality should be not too difficult to be learned, used, and remembered.

We have conducted a variety of empirical studies to determine the usage patterns of HFAs, their structure, and their associated help and learning mechanisms. All of these studies have led us to the identification of the qualitative relationships between usage patterns of HFAs as illustrated in Figure 4.

The ovals in Figure 3 represent users' knowledge about the system's concepts set. D_1 represents concepts that are well known, easily employed, and used regularly by a user. D_2 contains concepts known vaguely and used only occasionally, often requiring passive help systems. D_3 represents concepts users *believe* to exist in the system. The rectangle D_4 represents the functionality provided by the system. The "D_3 and not D_4" domain represents concepts in the user's mental model that they expect to exist, although they do not exist in the actual system. End-user modification and programming support is needed to empower users to add this functionality (Fischer and Girgensohn, 1990).

As the functionality of HFAs increases to $D_{4'}$, little is gained for users unless there are mechanisms to help them relate the additional functionality to their needs. Most users do not want to become technical experts – they just want to get their tasks done. The area of D_4 that is not part of D_3 is of specific interest to research in user modeling. This is system functionality, whose existence is unknown to users. For the "D_4 and not D_3" domain, information access (the user-initiated location of information when they perceive a need for an operation) is not sufficient, but information delivery (the system volunteering information that it inferred to be

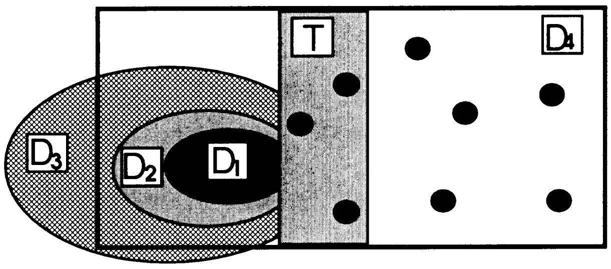


Figure 5. Functionality and its relevance to the task at hand

relevant to the users' task at hand) is required. Active help systems and critics are required to point out to users functionality that may be useful for their tasks and to help users avoid getting stuck on suboptimal plateaus.

Figure 4 shows usage patterns of HFAs without taking specific tasks of users into account. There is no reason for users to worry about additional existing functionality in D_4 if this functionality is not relevant to their tasks. However, if the system *does* provide functionality in D_4 related to users' tasks, it is desirable to avoid having users unable to perform the task or do so in a suboptimal or error-prone way because they do not know about this functionality. In Figure 5 the gray rectangle T represents the information that is relevant to the users' task at hand, and the dots represent different pieces of functionality. *Passive* support systems supporting information access can help users to explore pieces of functionality that are contained in D_3 and T, whereas *active* intelligent systems supporting information delivery are needed for the functionality contained in T and not in D_3. The functionality of *all* dots, including the ones in D_4 outside of T is often offered by specific push systems such as "Did You Know" (DYK) systems (Owen, 1986) or Microsoft's "Tip of the Day" (Roberts, 1989). This approach suffers from the problem that concepts get thrown at users in a decontextualized way.

3.2.1. *Expertise in HFA*

"Experts" (users who know everything about a system) no longer exist in HFAs. In HFAs, being an "expert" is at best an attribute of a specific context, rather than a personal attribute. The different spaces of expertise (determined by individual interest) are illustrated in Figure 6. In this *multi-kernel model*, $\{D_1, U_i\}$ means the area of functionality that is well known to a particular user U_i; for example: U_1 knows about the equation editor, U_2 knows about mail-merge functionality, U_3 uses a bibliography system for references, and U_4 is familiar with collaborative writing tools. This view provides a rationale why HFAs exist in this world: because designers need to write software for millions of users (at design time) for a large space of different tasks to be known only at use time.

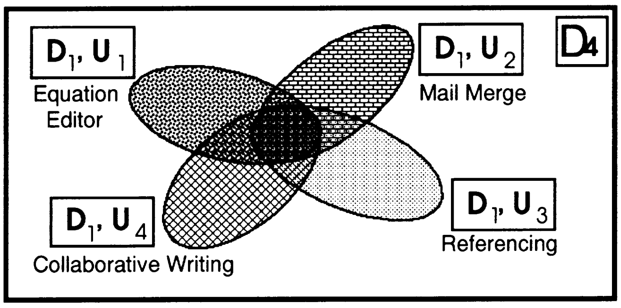
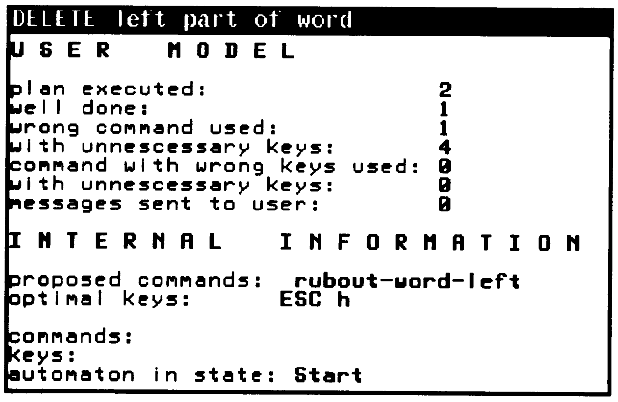


Figure 6. Distributed expertise in HFAs

HFAs create challenging *learning problems* that are representative for numerous complex systems. As illustrated with the above figures, nobody learns these systems completely, but users acquire some base functionality and learn additional functionality on demand. User-modeling techniques can effectively support *learning on demand* (Fischer, 1991) by helping users to identify opportunities to learn additional functionality relevant to their task at hand and to avoid people becoming stuck on suboptimal plateau. User modeling techniques based on logged user data can support the organization-wide learning of HFAs (Linton et al., 1999). Typical knowledge workers have become so deluged with information that they find it increasingly difficult to access the information they need – the sheer volume of irrelevant information makes it difficult to find information that *is* relevant. This challenge is particularly important in the context of organizational memory systems, because most of these systems will contain too much information for browsing to be effective. The *Invision* project was an early user modeling approach to use explicitly represented models of the knowledge and information needs of members of the organization (Kass & Stadnyk, 1992) to improve organizational communication and organizational learning.

3.3. Knowledge-based help systems

Our research related to user modeling has attempted to address the challenges created by HFAs. *Active help systems* (such as Activist (Fischer et al., 1985)), UNIX Consultant (Wilensky et al., 1984), and EUROHELP (Winkels et al., 1991)) were an early attempt to analyze the behavior of users and infer higher-level goals from low-level operations (Horvitz et al., 1998; Nardi et al., 1998). *Activist* is an active help system for an EMACS-like editor. It supported humans to incrementally learn more about pieces of functionality of relevance to their tasks (see Figure 5). It addressed the problem that humans often learn by receiving answers to questions that they have never posed or were not able to pose. Activist is able to deal with two different kinds of suboptimal behavior: (1) the user does not know a complex command and uses "suboptimal" commands to reach a goal, and (2) the user knows



```
DELETE left part of word
USER   MODEL

plan executed:                  2
well done:                      1
wrong command used:             1
with unnescessary keys:         4
command with wrong keys used:   0
with unnescessary keys:         0
messages sent to user:          0

INTERNAL   INFORMATION

proposed commands:  rubout-word-left
optimal keys:       ESC h

commands:
keys:
automaton in state: Start
```

Figure 7. An example of a specific plan specialist

the complex command but does not use the minimal key sequence to issue the command. A particular plan specialist (which deals with "deleting the left part of a word") is shown in Figure 7.

Similar to a human observer, Activist was able to handle the following tasks: (1) it recognizes what the user is doing or wants to do; (2) it evaluates how the user tries to achieve a goal; (3) it constructs a model (accumulating information over long usage periods) of the user based on the results of the evaluation task; and (4) it decides (dependent on the information in the model) *when* and *how* to interrupt.

The recognition and evaluation task in Activist was delegated to 20 different plan specialists. As an example of a user modeling system, Activist modeled its users on "actual observation" of actions rather than on inferring beliefs (McKeown, 1990). Contrary to the WEST system, Activist operated in a much more open-ended environment, creating the challenge to infer user goals from low-level interactions. The basic design philosophy of Activist allowed users to ignore the advice given by the system; if they ignored the information provided by a specific plan specialist repeatedly, it was turned off by Activist.

The INFOSCOPE system (Stevens, 1993) was a knowledge-based system that assisted users to locate interesting information in large poorly structured information spaces. Relying on agents that captured usage data, it created a user model to help users to locate personally meaningful information more easily.

3.4. Design environments and critiquing systems

HFAs, as argued above, came into existence as environments that are useful for a large number of different users (see Figure 6). In order to reduce their complexity, HFAs have often migrated to a collection of domain-oriented subsystems, each with their own templates, forms, and associated wizards, thereby enabling them to provide additional support for user modeling and assistance not available in more general systems.

In our own research, we have taken this approach further by developing *domain-oriented design environments* (Fischer, 1994). These are environments that model specific domains (such as computer networks, user interfaces, kitchens (Nakakoji, 1993), and voice dialog design (Sumner, 1995)) by allowing designers to engage in authentic tasks from their own respective work practices. Domain-oriented design environments allow computers to be "invisible" thus enabling designers to communicate with domain-specific concepts, representations, and tools. By bringing objects closer to the conceptual world of their users, the domain orientation of these environments makes HFAs more usable, more useful, and more learnable.

Domain-oriented design environments integrate a number of components relevant to user modeling:

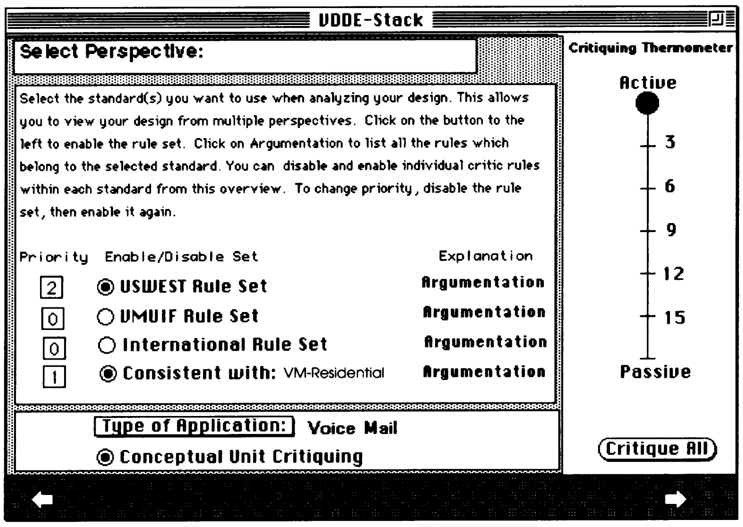
- They provide specification components (Nakakoji, 1993) that allow users to enrich the description of their tasks.
- They contain critiquing components (Fischer et al., 1998) that analyze and infer the task at hand in order to be able to detect and identify the potential for a design information need and then deliver task-relevant information for domain designers.
- They use the artifact (including its partial construction and partial specification) combined with domain knowledge contained in the design environment as an indication of the user's intentions, thereby providing an opportunity to infer high-level goals from simple user actions.

In the context of this research, we have developed principles and arguments for the trade-off between *adaptive* and *adaptable* approaches in user modeling (see Figure 8) (see Fischer (1993b); Oppermann (1994); Thomas (1996)). Based on the information provided at use time (see Figure 3) with specification components (Nakakoji, 1993), generic critics (defined at design time) can be *adaptively* refined to specific critics (Fischer et al., 1998) which are more sensitive to particular user needs. End-user mechanisms (Girgensohn, 1992) provide means for domain designers to *adapt* and evolve the systems to their specific needs.

Another example for user adaptation in domain-oriented design environments is shown in Figure 8. The figure shows a screen image from the *Voice Dialog Design Environment* (Sumner et al., 1997), which supports *adaptation mechanisms.* It allows users to select specific rule sets for critiquing and to determine the intervention strategy for the *intrusiveness* of the critics, ranging from active behavior (every step is immediately critiqued) through intermediate levels to passive behavior (users have to explicitly invoke the critiquing system). This adaptation mechanism provides users with the control to operate a critiquing systems in either an information delivery ("push") or information access ("pull") mode. Similar solutions to put the level of intrusiveness under the control of the users at use time rather than deciding it by designers at design time can be found in modern spelling correction programs. The need to customize and tailor critiquing systems to individual users'

Table 1. A comparison between adaptive and adaptable systems

	Adaptive	Adaptable
Definition	dynamic adaptation by the system itself to current task and current user	user changes (with substantial system support) the functionality of the system
Knowledge	contained in the system; projected in different ways	knowledge is extended
Strengths	little (or no) effort by the user; no special knowledge of the user is required	user is in control; user knows her/his task best; system knowledge will fit better; success model exists
Weaknesses	user has difficulty developing a coherent model of the system; loss of control; few (if any) success models exist (except humans)	systems become incompatible; user must do substantial work; complexity is increased (user needs to learn the adaptation component)
Mechanisms Required	models of users, tasks, and dialogs; knowledge base of goals and plans; powerful matching capabilities; incremental update of models	layered architecture; domain models and domain-orientation; "back-talk" from the system; design rationale
Application Domains	active help systems, critiquing systems, differential descriptions, user interface customization, information retrieval	information retrieval, end-user modifiability, tailorability, filtering, design in use



Select the standard(s) you want to use when analyzing your design. This allows you to view your design from multiple perspectives. Click on the button to the left to enable the rule set. Click on Argumentation to list all the rules which belong to the selected standard. You can disable and enable individual critic rules within each standard from this overview. To change priority, disable the rule set, then enable it again.

Priority	Enable/Disable Set	Explanation
2	● USWEST Rule Set	Argumentation
0	○ VMUIF Rule Set	Argumentation
0	○ International Rule Set	Argumentation
1	● Consistent with: VM-Residential	Argumentation

Type of Application: Voice Mail

● Conceptual Unit Critiquing

Figure 8. Adaptation mechanism to control different critiquing rule sets and different intervention strategies

objectives has also been explored in the domain of business graphs and use of color (Gutkauf, 1998).

Domain-oriented design environments pose a number of challenging problems for user modeling. Contrary to the WEST system discussed above, they model (open-ended) domains in support of self-directed learning and user-directed activities within a domain. They are able to exploit domain models for user modeling (Hollan, 1990). At design time (when the system is developed), domain models including generic critiquing knowledge and support for specification as well as facilities for end-user adaptations are provided. In these *open-ended systems* (for which the domain is determined at design time, but the tasks within the domain are undefined until use time and under the control of domain designers), the task in which users engage cannot be anticipated: it has to be inferred or articulated at use time. User modeling is a way to leverage critics to be not just relevant to the task at hand but be more responsive to the needs and the situation of an individual user. By having access to interactions the user has had in the past, the critic's advice can be specifically tailored (Mastaglio, 1990) so users are not bothered with information they already know or are not interested in.

4. Assessment of Progress

4.1. Addressing the challenges of HFAs

As the complexity of commercially available HFAs grows, and as we see more computational assistants (such as agents, advisors, coaches, and critics) appear in widely available commercial applications, an analysis and understanding of how people learn, work, and collaborate with and around HFAs will provide us with new requirements for the design of effective user modeling components. As argued before, tasks supported by HFAs are broad, and users do not know about all of the software features that could help them in principle and in some tasks that they may never attempt. In any HFA, there are functions for tasks used too infrequently by users to make it worthwhile for them to learn them and complex enough that they need to relearn how to perform them each time they try to accomplish the task. Most users do not want to become technical experts by learning in the abstract a large number of functions provided by HFAs – they just want to get their tasks done. Active help and critiquing that provide information relevant to the task at hand and relate it to a specific user's expertise and background knowledge, are important components to let users exploit the power of HFAs

4.2. Commercial applications: Microsoft's IntelliSense

Microsoft, with its IntelliSense software components (which first appeared in Office 97), started to tackle the problems of HFAs in its commercial applications. As briefly mentioned before, the "Tip of the Day" is a feature that tried to acquaint users with the functionality in D_4 (see Figure 4). Despite the possibility for interesting serendipitous encounters of relevant information, most users find this feature more annoying than helpful (and consequently, many people turn it off if they know how to do it), because a concept is thrown at them in a decontextualized fashion. Other features of the IntelliSense software were based on the Lumière project (Horvitz et al., 1998) that used Bayesian user models to infer a user's need by taking a user's background, actions, and queries into account.

Other commercial system components acting as critics, such as spelling and grammar checks, could be made considerably more useful by linking them with a user model. There are numerous software components that attempt to "understand" the context of an end-user's actions, or recognize the user's intent, and either automatically produce the correct result (e.g. AutoCorrect) or offer the assistance of wizards (e.g. for creating faxes or letters).

In addition to these adaptive components, these systems contain numerous features for adaptation (such as preferences and customization components) that support the *personalization of the software* by allowing users to control how the HFA behaves. Many HFAs also allow users to create additional features by supporting end-user programming with macros and embedded (simple programming or scripting) languages.

5. Future Challenges

User modeling in HCI faces a number of interesting challenges, including the following.

5.1. Increase the payoff of user modeling

There is little to be gained if expensive mechanisms are used to achieve minimal improvements in usability and usefulness (e.g. our empirical investigations have shown that few users, even trained computer scientists, take advantage of the MS-Word macro adaptation mechanisms). The payoff or utility of cognitive artifacts can be characterized by the quotient of "value/effort." To increase the payoff, we have two options: (1) increase the value by showing that future systems relying on user models are more usable and more useful; or (2) decrease the effort associated with creating a user model (for example, by exploiting usage data).

5.2. Differentiate between user modeling and task modeling

In many cases, we are not interested in user modeling in any general sense, but only in user performance and background knowledge with respect to tasks in a certain domain (see Figure 6). In this case, an adequate user model can be restricted to a small set of user attributes related to a specific task.

5.3. Support different modeling techniques

Many user modeling approaches failed because they relied too much on one specific technique. There is evidence (Thomas, 1996) that substantial leverage can be gained by *integrating modeling* (e.g. with specification components, with questionnaires) with *implicit modeling* (e.g. analyzing user performance on tasks, inferring the knowledge background and interests based on previous interactions) (Kass, 1991). This enriched synthesis can be further complemented by asking users (in otherwise open environments, such as HFAs and design environments) to solve specific problems in which the selection of the problem is driven by specific needs of the user modeling component.

5.4. Dealing with user models containing wrong, outdated, and inadequate information

User models represent a world that is outside the computational environment. The mapping of external information (particularly if we rely on inferred rather than observed behavior) to the internal model may be wrong to start with, but even under the assumption that it is an adequate representation at some point of time, it may become outdated by external changes of which the model is unaware (Allen, 1997). How, when, and by whom can a wrong user model be identified? Who will have the authority and the knowledge to change the model, and which modification mechanisms will be available to do so?

5.5. Develop criteria to judge the adequacy of user modeling in different domains

Assuming that user modeling is useful in some domains but not in others, which criteria do we have to distinguish these domains?

5.6. Capturing the larger context

Suchman (1987) argues convincingly that ‘‘interaction between people and computers requires essentially the same interpretive work that characterizes interaction between people, but with fundamentally different resources available to the participants. People make use of linguistic, nonverbal, and inferential resources in finding the intelligibility of actions and events, which are in most cases not available and not understandable by computers.’’ This raises the interesting challenges: (1) How can we capture the larger (often unarticulated) context of what users are doing (especially beyond the direct interaction with the computer system)? (2) How can we increase the ‘‘richness of resources’’ available for computer programs attempting user modeling to understand (what they are told about their users) and to infer from what they are observing their users doing (inside the computational environment and outside) (Horvitz et al., 1999)?

Ubiquitous computing (Weiser, 1993), embedded communication (Reeves, 1993), and usage data (Hill et al., 1992) make an attempt to reduce the unnecessary separation of computational artifacts from the physical objects they represent and from the discussions surrounding them (this separation created computational environments that are "deaf, blind, and quadriplegic agents" (Bobrow, 1991)). History and interaction patterns document how artifacts were developed and which actions and contributions individual users have made. Circumstantial indexing (Bolt, 1984), for example, is a powerful retrieval technique used by human collaborators that allows users to remember events in terms of things they did, not necessarily in terms of things that happened to objects.

5.7. User modeling and control

A consequence of any smart behavior of systems is that agents (humans or computers) can guess wrong and perform hidden changes that users do not like. Current systems often lack the possibility or at least the transparency for users to turn off these "smart" features, which can get more in the way than help. As argued above, systems, even smart ones, are aware of only a fraction of the total problem-solving process their human partners undergo (Hollan, 1990), and they cannot share an understanding of the situation or state of problem-solving of a human (Suchman, 1987). Whereas these drawbacks of smart systems may be only annoying in HFAs such as word processors, they are unacceptable in other collaborative human–computer systems, such as airline cockpit computers serving as intelligent agents. Billings (1991) argues convincingly that in computerized cockpit design each intelligent agent in a human–computer system must have knowledge of the intent and the rationale of the actions of the other agents. To avoid these drawbacks, intelligent systems should provide malleable tools (Fischer, 1993b) that empower rather than diminish users, giving them control over tasks necessary for everyday life (Kay, 2001). There are situations in which we desire automation and intelligence (for example, few people will have the desire to compile their programs themselves) – but the decision as to what should be automated and what not should be under the control of the people affected by the system (Shneiderman and Maes, 1997).

5.8. Privacy and user models

We live in a world where more and more events take place and are tracked in some computational environment and recorded in a user model, just to name a few examples: telephone calling cards, shopping cards at supermarkets, book ordering at electronic book stores, websites visited, active badges worn by humans (Harper et al., 1992). Numerous organizations compile user models of our behavior and actions – and there is the great danger that this information can be misused. It will be a major challenge to find ways to avoid misuses, either by not allowing companies to collect this information at all or by finding ways that the individual users have control over these user models.

6. Conclusions

The ultimate objective of user modeling is that it has to be done for the benefit of users. Past research has shown that there is often quite a difference between modeling certain aspects of a user's work and behavior, and applying this knowledge for the benefit of the user. User modeling, particularly in its impact and value for future collaborative human–computer systems, has traveled a long and winding road (Fischer, 1999). This is true not only for the past, but equally true for the future. Many interesting and challenging problems are ahead of us: how to understand the trade-offs, the promises, and the pitfalls (1) between adaptive and adaptable systems; (2) between information delivery (''push'') and information access (''pull'') technologies; (3) between contextualized information representations and serendipity; and (4) how to move from our current desk-top environments to web-based environments. Hopefully, this road will lead us to new ideas and new insights in the design of future human-centered systems supported by adequate user modeling techniques.

Acknowledgments

I would like to thank the members of the Center for LifeLong Learning and Design (L^{3}D) who have contributed to the ideas and perspectives expressed in this paper. A special *thank you* goes to my former students (Taro Adachi, Thomas Mastaglio, Kumiyo Nakakoji, Curt Stevens, and Tamara Sumner at CU-Boulder, and Bernd Gutkauf and Christoph Thomas in Germany) who, in their Ph.D. and Master theses, explored important issues in user modeling as they occur in high-functionality applications, domain-oriented design environments, and critiquing systems.

The current version of this paper benefited greatly from valuable and important feedback from Eric Horvitz (Microsoft Research) and from Peter Brusilovsky, Judy Kay, Constantine Stephanidis, and Ingrid Zukerman, co-authors of this special issue.

The research was supported by (1) the National Science Foundation, Grants REC-9631396 and IRI-9711951; (2) Software Research Associates, Tokyo, Japan; (3) PFU, Tokyo, Japan; and (4) the Coleman Foundation, San Jose, CA.

References

Adachi, T.: 1998, *Utilization of Usage Data to Improve Organizational Memory*, M.S. Thesis, Department of Computer Science, University of Colorado at Boulder, Boulder, CO.

Allen, R. B.: 1997, Mental Models and User Models. In: M. G. Helander, T. K. Landauer, and P. V. Prabhu (eds.), *Handbook of Human-Computer Interaction, Volume 1*, Elsevier Science B.V., Amsterdam, pp. 49–63.

Billings, C. E.: 1991, *Human-Centered Aircraft Automation: A Concept and Guidelines*, NASA Technical Memorandum, Report No. 103885, NASA Ames Research Center.

Bobrow, D. G.: 1991, Dimensions of Interaction, *AI Magazine*, **12**(3), 64–80.

Bolt, R. A.: 1984, *The Human Interface*, Lifetime Learning Publications, Belmont, CA.

Burton, R. R. and Brown, J. S.: 1982, An investigation of computer coaching for informal learning activities. In: D. H. Sleeman and J. S. Brown (eds.), *Intelligent Tutoring Systems*, Academic Press, London – New York, pp. 79–98.

Card, S. K., Moran, T. P., and Newell, A.: 1983, *The Psychology of Human-Computer Interaction*, Lawrence Erlbaum Associates, Inc., Hillsdale, NJ.

Computer Science and Technology Board – National Research Council (ed.): 1997, *More Than Screen Deep: Toward Every-Citizen Interfaces to the Nation's Information Infrastructure,* National Academy Press, Washington, D.C.

Fischer, G.: 1991, Supporting learning on demand with design environments. In: L. Birnbaum (ed.), *International Conference on the Learning Sciences (Evanston, IL),* Association for the Advancement of Computing in Education, pp. 165–172.

Fischer, G.: 1993a, Beyond human computer interaction: Designing useful and usable computational environments. In: *People and Computers VIII: Proceedings of the HCI'93 Conference (Loughborough, England)*, Cambridge University Press, Cambridge, UK, pp. 17–31.

Fischer, G.: 1993b, Shared knowledge in cooperative problem-solving systems – integrating adaptive and adaptable components. In: M. Schneider-Hufschmidt, T. Kuehme, and U. Malinowski (eds.), *Adaptive User Interfaces – Principles and Practice*, Elsevier Science Publishers, Amsterdam, pp. 49–68.

Fischer, G.: 1994, Domain-oriented design environments, *Automated Software Engineering*, **1**(2), 177–203.

Fischer, G.: 1999, User modeling: The long and winding road. In: J. Kay (ed.), *Proceedings of UM99: User Modelling Conference (Banff, Canada)*, Springer Verlag, Wien New York, pp. 349–355.

Fischer, G. and Girgensohn, A.: 1990, End-user modifiability in design environments. In: *Human Factors in Computing Systems, (CHI'90) (Seattle, WA)*, ACM, New York, pp. 183–191.

Fischer, G., Lemke, A. C., Mastaglio, T., and Morch, A.: 1991, The role of critiquing in cooperative problem solving, *ACM Transactions on Information Systems*, **9**(2), 123–151.

Fischer, G., Lemke, A. C., and Schwab, T.: 1985, Knowledge-based help systems. In: L. Borman and B. Curtis (eds.), *Proceedings of CHI'85 Conference on Human Factors in Computing Systems*, ACM, New York, pp. 161–167.

Fischer, G., Nakakoji, K., Ostwald, J., Stahl, G., and Sumner, T.: 1998, Embedding critics in design environments. In: M. T. Maybury and W. Wahlster (eds.), *Readings in Intelligent User Interfaces*, Morgan Kaufmann, San Francisco, pp. 537–561.

Fischer, G. and Stevens, C.: 1987, Volunteering information – enhancing the communication capabilities of knowledge-based systems. In: H. Bullinger and B. Shackel (eds.), *Proceedings of INTERACT'87, 2nd IFIP Conference on Human-Computer Interaction (Stuttgart, FRG)*, North-Holland, Amsterdam, pp. 965–971.

Girgensohn, A.: 1992, *End-User Modifiability in Knowledge-Based Design Environments*, Ph.D. Dissertation, Department of Computer Science, University of Colorado at Boulder, Boulder, CO.

Gutkauf, B.: 1998, *Improving Design and Communication of Business Graphs through User Adaptive Critiquing*, Ph.D. Dissertation, Mathematics and Computer Science, Universität-GH Paderborn, Paderborn, Germany.

Harper, R., Lamming, M. G., and Newman, W. M.: 1992, Locating systems at work: Implications for the development of active badge applications, *Interacting with Computers,* **4**(3), 343–363.

Helander, M. G., Landauer, T. K., and Prabhu, P. V. (Eds.): 1997, *Handbook of Human-Computer Interaction*, (Second, Completely Revised ed.), Elsevier Science Ltd., Amsterdam.

Henderson, A. and Kyng, M.: 1991, There's no place like home: Continuing design in use. In: J. Greenbaum and M. Kyng (eds.), *Design at Work: Cooperative Design of Computer Systems*, Lawrence Erlbaum Associates, Inc., Hillsdale, NJ, pp. 219–240.

Hill, W. C., Hollan, J. D., Wroblewski, D., and McCandless, T.: 1992, Edit wear and read wear. In: P. Bauersfeld, J. Bennett, and G. Lynch (eds.), *Proceedings of CHI'92 Conference on Human Factors in Computing Systems*, ACM, New York, pp. 3–9.

Hollan, J. D.: 1990, User models and user interfaces: A case for domain models, task models, and tailorability. In: *Proceedings of AAAI-90, Eighth National Conference on Artificial Intelligence*, AAAI Press/The MIT Press, Cambridge, MA, p. 1137.

Horvitz, E.: 1999, Principles of mixed-initiative user interfaces. In: *Human Factors in Computing Systems, CHI'99 (Pittsburgh, PA)*, ACM, New York, pp. 159–166.

Horvitz, E., Breese, J., Heckerman, D., Hovel, D., and Rommelse, K.: 1998, The Lumiere Project: Bayesian user modeling for inferring the goals and needs of software users. In: *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (Madison, WI)*, Morgan Kaufmann, San Francisco, pp. 256–265.

Horvitz, E., Jacobs, A., and Hovel, D.: 1999, Attention-sensitive alerting. In: *Proceedings of UAI '99, Conference on Uncertainty and Artificial Intelligence (Stockholm, Sweden)*, Morgan Kaufmann, San Francisco, pp. 305–313.

Kass, R.: 1991, Building a user model implicitly from a cooperative advisory dialog, *User Modeling and User-Adapted Interaction* **1**(3), 203–258.

Kass, R. and Stadnyk, I.: 1992, Using user models to improve organizational communication. In: *Proceedings of 3rd International Workshop on User Modeling (UM'92)*, The German Research Center for Artificial Intelligence, Dagstuhl, Germany, pp. 135–147.

Kay, J.: 2001, Learner control, *User Modeling and User-Adapted Interaction* **11**(1–2), 111–127 (this issue).

Kobsa, A. and Wahlster, W. (eds.): 1989, *User Models in Dialog Systems*, Springer-Verlag, New York.

Linton, F., Joy, D., and Schaefer, H.-P.: 1999, Building user and expert models by long-term observation of application usage. In: J. Kay (ed.), *Proceedings of UM99: User Modelling Conference (Banff, Canada)*, Springer Verlag, Wien New York, pp. 129–138.

Mackay, W. E.: 1991, Triggers and barriers to customizing software. In: S. P. Robertson, G. M. Olson, and J. S. Olson (eds.), *Proceedings of CHI'91 Conference on Human Factors in Computing Systems*, ACM, New York, pp. 153–160.

Mastaglio, T.: 1990, *A User-Modelling Approach to Computer-Based Critiquing*, Ph.D. Dissertation, Department of Computer Science, University of Colorado at Boulder, Boulder, CO.

Maybury, M. T. and Wahlster, W.: 1998, *Readings in Intelligent User Interfaces*, Morgan Kaufmann, San Francisco.

McKeown, K. R.: 1990, User modeling and user interfaces. In: *Proceedings of AAAI-90*, Eighth National Conference on Artificial Intelligence, AAAI Press/The MIT Press, Cambridge, MA, pp. 1138–1139.

Nakakoji, K.: 1993, *Increasing Shared Understanding of a Design Task Between Designers and Design Environments: The Role of a Specification Component*, Ph.D. Dissertation, Department of Computer Science, University of Colorado at Boulder, Boulder, CO.

Nardi, B. A., Miller, J. R., and Wright, D. J.: 1998, Collaborative, programmable intelligent agents, *Communications of the ACM*, **41**(3), 96–104.

Oppermann, R. (ed.): 1994, *Adaptive User Support*, Lawrence Erlbaum, Hillsdale, New Jersey.

Owen, D.: 1986, Answers first, then questions. In: D. A. Norman and S. W. Draper (eds.), *User-Centered System Design, New Perspectives on Human-Computer Interaction*, Lawrence Erlbaum Associates, Inc., Hillsdale, NJ, pp. 361–375.

Reeves, B. N.: 1993, *The Role of Embedded Communication and Artifact History in Collaborative Design*, Ph.D. Dissertation, Department of Computer Science, University of Colorado at Boulder, Boulder, CO.

Rich, E.: 1989, Stereotypes and user modeling. In: A. Kobsa and W. Wahlster (eds.), *User Models in Dialog Systems*, Springer-Verlag, New York, pp. 35–51.

Roberts, R. M.: 1989, *Serendipity: Accidental Discoveries in Science*, John Wiley & Sons, Inc., New York.

Shneiderman, B. and Maes, P.: 1997, Direct manipulation vs. interface agents, *Interactions (ACM)*, **4**(6 (Nov)), pp. 42–61.

Stephanidis, C.: 2001, Adaptive techniques for universal access, *User Modeling and User-Adapted Interaction* **11**(1–2), 159–179 (this issue).

Stevens, C.: 1993, *Helping Users Locate and Organize Information*, Department of Computer Science, University of Colorado, Boulder, CO.

Suchman, L. A.: 1987, *Plans and Situated Actions*, Cambridge University Press, Cambridge, UK.

Sumner, T.: 1995, *Designers and Their Tools: Computer Support for Domain Construction*, Ph.D. Dissertation, Department of Computer Science, University of Colorado at Boulder, Boulder, CO. Available at: http://kmi.open.ac.uk/~tamara/thesis.html.

Sumner, T., Bonnardel, N., and Kallak, B. H.: 1997, The cognitive ergonomics of knowledge-based design support systems. In: S. Pemberton (ed.), *Proceedings of CHI 97 Conference on Human Factors in Computing Systems*, ACM/Addison-Wesley, pp. 83–90.

Terveen, L. G.: 1995, An overview of human–computer collaboration, *Knowledge-Based Systems Journal, Special Issue on Human-Computer Collaboration*, **8**(2–3), 67–81.

Thomas, C. G.: 1996, *To Assist the User: On the Embedding of Adaptive and Agent-Based Mechanisms*, R. Oldenburg Verlag, München/Wien.

Weiser, M.: 1993, Some computer science issues in ubiquitous computing, *Communications of the ACM*, **37**(7), 75–83.

Wilensky, R., Arens, Y., and Chin, D.: 1984, Talking to UNIX in English: An overview of UC, *Communications of the ACM*, **27**(6), 574–593.

Winkels, R., Breuker, J., and denHaan, N.: 1991, Principles and practice of knowledge representation in EUROHELP. In: L. Birnbaum (ed.), *International Conference on the Learning Sciences (Evanston, IL)*, Association for the Advancement of Computing in Education, Charlottesville, Virginia, pp. 442–448.

Author's Vita

Gerhard Fischer is Professor of Computer Science, a fellow of the Institute of Cognitive Science, and the director of the Center for Lifelong Learning and Design (L3D) at the University of Colorado at Boulder. His research interests include education and computers (including lifelong learning, learning on demand, and collaborative learning), human–human and human–computer collaboration, cognitive science, artificial intelligence, (software) design, and domain-oriented design environments.

More information about the (L^3D) center can be found at:
http://www.cs.colorado.edu/~l3d/