Adaptive User Interfaces
M. Schneider-Hufschmidt, T. Kühme & U. Malinowski (Editors)

Shared Knowledge in Cooperative Problem-Solving Systems - Integrating Adaptive and Adaptable Components

Gerhard Fischer

University of Colorado at Boulder
Boulder, USA

ABSTRACT

Integrated, domain-oriented, knowledge-based design environments are examples of cooperative problem-solving systems relying on shared knowledge. Research goals pursued in the context of design environments are to support human problem-domain communication, to make information relevant to the task at hand, and to tailor information to a specific user or class of users.

The shared knowledge between a user and a system will not be static, but it will increase and change over time. There are two major ways that this can be achieved: by making systems *adaptable* (e.g., by supporting end-user modifiability), and *adaptive* (e.g., systems act differently based on a model of a specific task situation or a specific user).

We have developed prototypes of design environments that (1) demonstrate the need for shared knowledge, (2) support the incremental growth of shared knowledge, and (3) use the shared knowledge to make the interaction more user-specific and more task-oriented. Adaptable and adaptive mechanisms are used to achieve these goals.

ACKNOWLEDGMENTS

The author would like to thank the members of the Human-Computer Communication group at the University of Colorado, who contributed to the conceptual framework and the systems discussed in this paper. The research was supported in part by the National Science Foundation under grants No. IRI-9015441, MDR-9253425, and CDA-8922510, by the NYNEX Science and Technology Center, and by Software Research Associates.

1 COOPERATIVE PROBLEM-SOLVING SYSTEMS

Cooperative problem-solving systems [Stefik 86, Hill 89, Fischer 90] are knowledge-based environments supporting users in a symbiotic relationship to generate a product of

their common effort. Knowledge-based systems can be designed to interact with their users in a cooperative fashion using several different interaction paradigms: doing, deciding, advising, tutoring, and critiquing. The main emphasis of our work has been to augment and empower human designers with domain-oriented design environments [Fischer 92] containing an embedded critiquing component [Fischer 91a].

Models [Norman 82] are of greater importance in cooperative problem-solving systems than in autonomous systems because the problem-solving activity and knowledge is shared by the cooperating agents. Two models are of special interest for shared knowledge systems [Fischer 91a]:

- $\mathbf{M_1}$: the users' models (the models that users have of systems and tasks), and
- $\mathbf{M_2}$: the systems' models (the models that systems have of users and tasks).

Cooperative problem-solving systems are *designed* systems. Comprehending designed systems requires an understanding of the goals, functions, and adaptive capabilities for which they can be used. The models associated with these systems are part of the design (i.e., they have to be designed too), and they can and should provide important requirements for the design.

High-Functionality Systems. Cooperative problem-solving systems require high-functionality systems for their realization, creating the following dilemma: on the one hand, these are systems where good models are most urgently needed; but on the other hand, it is unclear how these systems can be designed so users will be able to build models for them. Models for high-functionality computer systems cannot be deduced merely from experience because there are too many experiences to go through [Norman 86]. Learning complex systems is an incremental, indefinite process requiring an understanding of how users increase their knowledge and understanding of them in naturalistic settings over long periods of time.

High-functionality systems confront users with too much information. The challenge is to make the information relevant to the task at hand - i.e., delivering the right knowledge, in the context of a problem or a task, at the right moment for a human professional to consider [Fischer et al. 93]. Making information relevant to the task at hand poses many challenges for the design of interactive computer systems and it sets computer systems truly apart from other technologies (e.g., an example discussed in [Norman 93] is the printed version of the Official Airline Guide compared to (1) the electronic access to it allowing the display of results in a variety of ways, and (2) FlightFax providing schedule and fare information customized to a person's itinerary and delivering it with fax machines).

To develop better design requirements for high-functionality systems, usage patterns of them (as shown in Figure 1) provide important insights [Draper 84, Fischer 91a]. This qualitative analysis of users' knowledge about complex systems reveals two interesting findings:

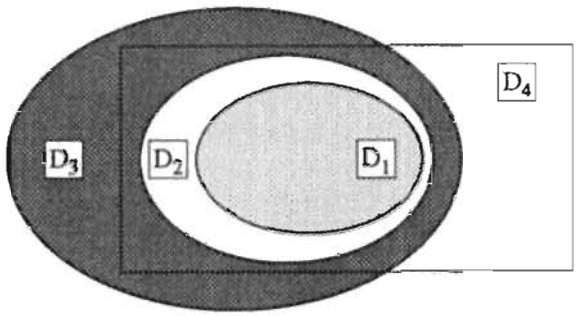


Figure 1: Levels of System Usage

In this figure the rectangle represents information embodied in a system (the system image) and ovals represent user knowledge about the system's information space.

D_1: The subset of concepts stored in the system's information repository that users know well and can use easily without the need for reference material.

D_2: The subset of concepts that users know vaguely and use occasionally. Users do not have complete understanding of the concepts, often requiring them to look the information up in manuals, etc.

D_3: The set of concepts that the user believes exist in the system. Note that some of the concepts lie outside of the actual information space.

D_4: The full set of concepts stored in the information repository of the system.

- The users' model of the system contains concepts that do not belong to the system (the part of D_3 that is not part of D_4).
- There are system parts of which users are unaware (the part of D_4 that is not part of D_3).

The former issue requires facilities assisting users in incrementally bringing their $\mathbf{M_1}$-type models closer in accordance with the actual system. To address the latter issue, intelligent support systems are needed that rely on $\mathbf{M_2}$-type models pointing out to users existing functionality that may be useful for their tasks.

M_1: The Users' Models of Systems. A user's model of a complex system is a cognitive construct that describes a user's understanding of a particular content domain in the world. These models are formed by experience, self-exploration, training, instruction, observation, and accidental encounters. In systems that operate at the "human-computer communication" level, the model will be centered around the properties of a computer system. An advantage of this type of model (representing a general computational environment) is that it is uniform across domains. In systems that operate at the "human problem-domain communication" level (giving users the feeling that they interact with concepts and representations drawn from the problem rather than from the computer domain [Fischer & Lemke 88]), users are able to form models using concepts much more closely related to an application domain.

M_2: The Systems' Models of Users and Tasks. There are a number of efforts to incorporate models of users into knowledge-based systems [Rich 83, Clancey 86, Kass & Finin 87, Fain-Lehman & Carbonell 89, Chin 89, Kobsa & Wahlster 89]. In our own research, we have investigated systems' models of users in connection with active help systems [Fischer et al. 85] and critics [Fischer 87, Fischer et al. 91a]. **M_2**-type models for critic systems pose specific demands. Unlike tutorial systems, which can track a user's expertise over a path of instruction, computer-based critics must work with users having a variety of background experiences. To operate effectively, critics should have a model of the task space in which the users operate. Having an adequate model, systems would allow for the (1) customization of explanations [Moore 89, Fischer et al. 90] so they cover exactly what users need to know; (2) provision of differential descriptions of new concepts in relationship to known concepts; (3) presentation of information through user-specific filters focusing on the parts that seem to be most relevant for a user [Fischer & Nakakoji 91]; and (4) they would keep active systems quiet most of the time [Fischer et al. 85].

Critics as Embedded System Components in Cooperative Problem-Solving Systems. Critiquing systems were first developed as stand-alone systems [Fischer 87]. Our current prototypes demonstrate that they are more powerful as embedded systems: critiquing is used as an interaction technique within integrated, domain-oriented design environments [Fischer et al. 91a, Fischer et al. 93]. The target audience for such systems is knowledge workers in application domains. Most domains have grown so complex that no single person can be considered an expert familiar with all aspects of the domain [Draper 84]. Support tools must include explanations of domain-specific knowledge and how this knowledge can assist users by critiquing their work. Experimental use of these prototypes demonstrated that in order for these systems to be truly cooperative, they must be tailored to the specific tasks and knowledge backgrounds of individual users.

As cooperative problem-solving systems move away from fine-grained analysis of simple interactions with a computer toward a focus on skilled users of high functionality systems, an essential characteristic of these systems will be their ability to adapt to their users. Systems' models of users in support of cooperative problem solving need to be dynamic, per-

sistent, and domain-oriented. Achieving these goals requires co-adaptive systems [Mackay 92] that transcend system architectures based on a static user and static software environment.

The Desirability of Malleable Systems. Malleable systems are desirable for the following reasons: (1) to support mutual intelligibility (reciprocal recognizability of our actions, enabled by common conventions for the expression of intent, and shared knowledge about typical situations), (2) to support communicative economy (if the premises or rationale of an action can be assumed to be shared, they can be left unspoken), and (3) to achieve that tools and artifacts become ready-to-hand and invisible allowing users to communicate more directly with the task.

2 INTEGRATED, DOMAIN-ORIENTED, KNOWLEDGE-BASED DESIGN ENVIRONMENTS

Based on a number of design efforts in specific domains (e.g., kitchen design [Fischer et al. 89], user interface design [Lemke & Fischer 90], and computer network design [Fischer et al. 92]), we have developed a general architecture for integrated, domain-oriented, knowledge-based design environments (see Figure 2).

Our architecture currently consists of five components and three integrating mechanisms. The five components are:

- A *construction kit* [Fischer & Lemke 88] is the principal medium for modeling a design. It provides a palette of domain concepts and supports construction using direct manipulation and electronic forms.
- An *argumentative hypertext system* [Fischer et al. 91b] contains generic issues, answers, and arguments about the design domain. Users can annotate and add argumentation as it emerges during the design process.
- A *catalog* [Fischer et al. 92] is a collection of prestored designs illustrating the space of possible designs in the domain, and supporting reuse and case-based reasoning [Riesbeck & Schank 89]. By serving as a group memory, catalogs support long-term indirect communication among groups of designers. Group memories contain a collection of shared information repositories [Resnick 91] containing a cumulative record of rationale, solution components, information about prior projects, and other information resources for collaboration. There are two crucial issues concerning such a memory: (1) how information gets into the memory and how it accumulates, and (2) how information in the memory is made available to the individual designer.
- A *specification component* [Fischer & Nakakoji 91] allows designers to describe characteristics of the design they have in mind. The specifications are expected to be modified and augmented during the design process, rather

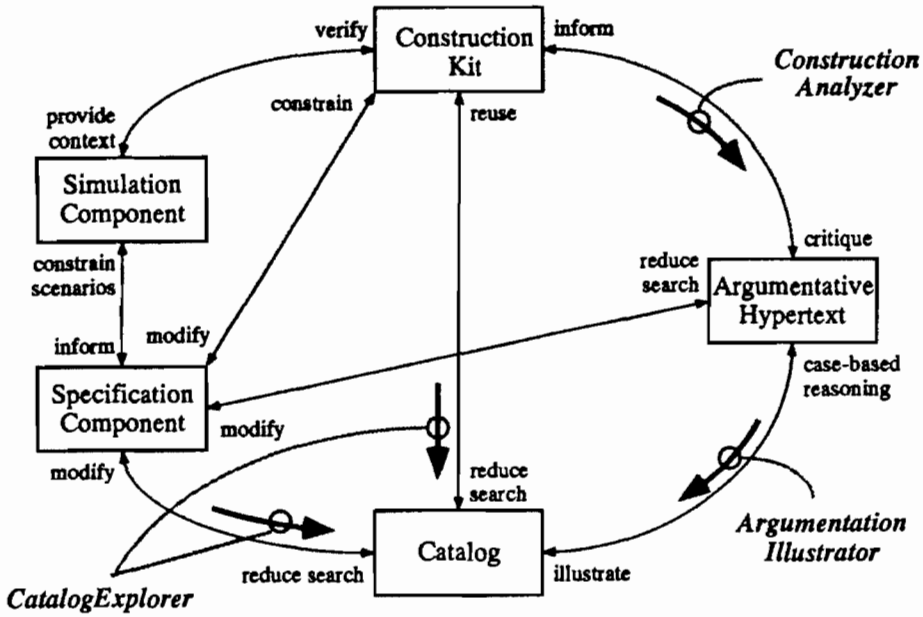


Figure 2: A Multifaceted Architecture

The components of the multifaceted architecture. The links between the components are crucial for exploiting the synergy of the integration.

than to be fully articulated at the beginning. They are used to retrieve design objects from the catalog and to filter information in the hypertext.

- A *simulation component* allows designers to carry out "what-if" games - that is, to simulate various usage scenarios involving the artifact being designed.

At each stage during the design process, the partially completed design embedded in the design environment serves as a stimulus suggesting to users what they should attend to next. To exploit the full power of the multifaceted architecture, the individual components need to be integrated. Currently the architecture supports the following linking mechanisms (see Figure 2):

- CONSTRUCTION ANALYZER. Users need support for construction, argumentation, and perception of breakdowns. Breakdowns are identified by the CONSTRUCTION ANALYZER operating as a critiquing system [Fischer et

al. 91b]. The firing of a critic signals a breakdown and provides entry into the argumentative hypermedia system at which the corresponding argumentation is located. Accessing useful knowledge has as a prerequisite that the demand be noticed by the user, requiring that the situation talks back [Schoen 83]. For users who do not have extensive experience in the domain, the situation is often mute unless the environment has a component that speaks up and points out issues that the designer may otherwise not have considered. Critics can fulfill this role. Critics point out suboptimal aspects of the artifact and know the places where the corresponding issues are discussed in the argumentation component [Fischer et al. 91b].

- ARGUMENTATION ILLUSTRATOR. The explanation given in the form of argumentation is often highly abstract and conceptual. Concrete design examples matching the explanation help users to understand the concept. The ARGUMENTATION ILLUSTRATOR helps users to understand information given in the argumentative hypertext by finding a catalog example that illustrates the concept.

- CATALOGEXPLORER. CATALOGEXPLORER helps users search the catalog space according to the task at hand [Fischer & Nakakoji 91]. It retrieves design examples similar to the current construction and specification. The catalog and the CATALOGEXPLORER are used to explore the roles of examples.

3 WHY DESIGN ENVIRONMENTS NEED TO BE ADAPTIVE AND ADAPTABLE

Adaptive systems change themselves based on the user's behavior [Fischer et al. 85]. An adaptive system must contain models of the domain, of the task, and/or of the users to adapt appropriately. Adaptive systems have among their goals (1) to filter information in a user- and task-specific way (so the "knowledge in the head" of a specific user is complemented naturally by the "knowledge in the world" offered by the system [Norman 93]), and (2) to present to users information of which they are not aware of (represented by the part of D_4 that is not part of D_3 in Figure 1) thereby supporting learning on demand. Adaptable systems are systems that can be modified by users in non-obvious ways (e.g., beyond the choosing of certain parameter settings [Henderson & Kyng 91]). The goals of adaptable systems include (1) making the system fit new requirements by adding or changing knowledge structures, and (2) evolving seeds by creating functional enhancements [Fischer et al. 92].

Adaptable systems allow users to modify the systems while working with them (examples for adaptable systems are Microsoft Word, EMACS [Stallman 81], NoteCards [Trigg et al. 87], BUTTONS [MacLean et al. 90], and OBJECT-LENS [Lai & Malone 88]). Adaptable

systems allow users to change the domain model (i.e., D_4 in Figure 1). A specific approach making systems adaptable is supporting end-user modifiability [Fischer & Girgensohn 90, Girgensohn 92].

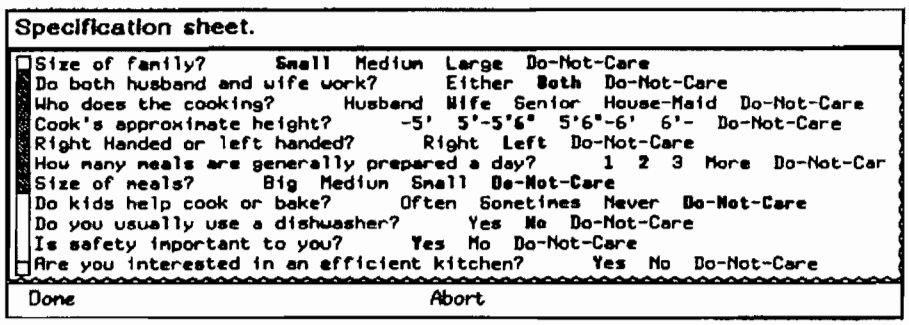
A Comparison Between Adaptive and Adaptable Systems. Adaptable and adaptive systems can and should be used complementarily. Figure 3 gives a high-level comparison between adaptive and adaptable systems.

	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 • system knowledge will fit better • success models exist
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 (users need to learn and know to interact with 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 • human problem-domain communication • "back-talk" from the system • design rationale
Application Domains	• active help systems • critiquing systems • differential descriptions • user interface customization	• end-user modifiability • tailorability • information filtering • design in use

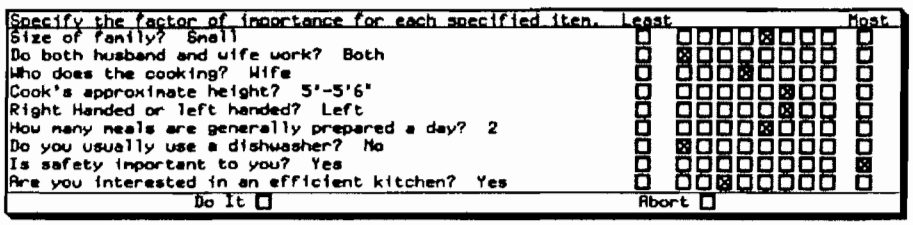
Figure 3: A Comparison Between Adaptive and Adaptable Systems

Why Design Environments Need to Be Adaptive. Design environments are high-functionality systems. The need for adaptive mechanisms is illustrated by the following example: the critiquing component of our systems encodes generic design knowledge about domains (e.g., such as "the stove should be in front of a window" or "the work-triangle should be less than 23 feet" in the kitchen design domain [Fischer et al. 89]). Without a

specification component, the system will be stuck with generic advice, and it will be unable to respond to user-specific situations, such as "a person in the household is only 5 feet tall" or "the family has a large number of small children"). If the system has this knowledge, critic messages and example selections can be made adaptive to the specific design situation. To support this adaptivity, we have developed a specification component [Fischer & Nakakoji 91] allowing users to describe the unique features of their design situation to the system. Specification components are explicit knowledge acquisition components in support of making systems more responsive to the task at hand (see Figure 4).



(a): Specification Sheet



(b): Weighting Sheet for the Specification

Figure 4: Specification Component of a Design Environment for

(a): The Specify command in CATALOGEXPLORER provides a specification sheet in the form of a questionnaire.

(b): After specification, users weigh the importance of each specified item.

Why Design Environments Need to Be Adaptable. Design environments must be adaptable because (1) human knowledge is tacit [Polanyi 66] (i.e., humans know more than they can say), and (2) the domain that the environments model changes over time.

End-user modifiable systems support their users in modifying systems according to their own needs. End-user modifiability allows users to tailor a system to pursue additional tasks, to have different preferences, and to adapt the system to changing needs over time in the real world. The intended users of end-user modifiable systems are knowledgeable in the application domain but unable or unwilling to modify a system on the programming language level.

Malleable Systems: Integrating Adaptive and Adaptable Components. Our design environments integrate adaptive and adaptable components in a variety of ways. The construction situation and the specification component are sources of information for adaptive features (e.g., influencing the set of active critics, and making information in the catalog and in the argumentation component relevant to the task at hand). Figure 5 illustrates that system architectures integrating adaptive and adaptable components are based on shared decision making requiring shared knowledge.

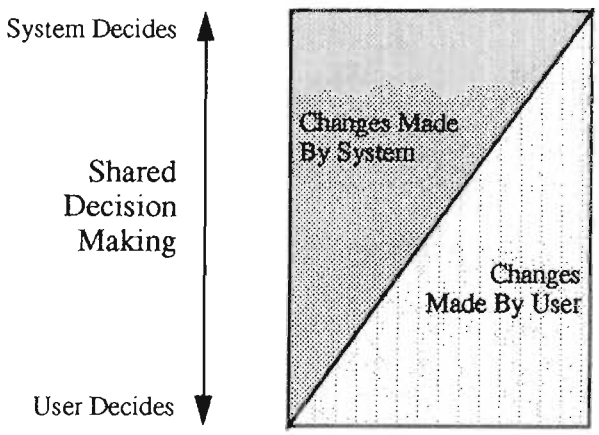


Figure 5: Integrating Adaptable and Adaptive Components

There is a broad spectrum of shared decision making between purely adaptive and purely adaptable systems in which users and system components contribute to the modification of a system.

The integration can be illustrated by showing how a standard critiquing system can be extended to a conditional one. A critiquing system that criticizes all designs based on the same standards is based on standard critics [Fischer et al. 93]. Design domains often have a basic set of rules that all artifacts in that domain should follow (e.g., in kitchen design [Fischer et al. 89] there are building codes, safety regulations, and functional principles; and in computer network design [Fischer et al. 92] there are standards established by committees). Standards that apply to all designs in a domain are important for designers to

understand and follow. Novices especially need to be reminded when their design violates the design standards, but even for experts this type of support is necessary in situations where there is too much detail for one person to process.

Although standard critics are good at enforcing a set of rules that may be applied to all designs for a domain, they do not fully support design processes. Design theorists [Schoen 83, Rittel 84] tell us that each design project should be seen as unique. It is the responsibility of the designer to understand the unique characteristics of the design situation and to formulate a solution that address the unique characteristics.

Obeying general rules and design standards is necessary but not sufficient for good design. Design environments must support the designer in seeing the situation at hand as unique. Domain standards can help constrain designs, but at the same time standards alone do not determine a design solution. For interesting design domains, generic design rules play only a part in the final product of design. Conditional critics allow a design environment to evaluate design situations in accordance with partial specifications [Fischer et al. 93]. The partial specification represents a set of goals articulated by the user. Each specification item corresponds to a set of critics, which detect design situations relevant to that specification item. The set of specification items chosen by the designer determines which critics are active. Only active critics participate in the evaluation of design situations.

The partial specification is a resource for both the system and the designer (and is thereby an important part of the shared knowledge): (1) it allows the system to generate design-specific (rather than domain-specific) critiquing, and (2) it allows the designer to understand the design in terms of its unique characteristics rather than its common ones.

Conditional critics can detect design situations where one specification item conflicts with another, creating trade-off situations. To resolve a trade-off between conflicting goals, the designer must decide which goal (articulated as one specification item) is more important. To support this type of reasoning, specification items can be weighted. Being able to quickly manipulate the specification allows the designer to investigate the implications of different priority schemes. Support for understanding trade-offs is vital for complex problems where goals and priorities cannot be known a priori.

Machine learning can be seen as another approach for integrating adaptive and adaptable systems. It can be defined operationally to mean the ability to perform new tasks that could not be performed before or perform old tasks better as a result of changes produced by the learning process [Carbonell 89]. Machine learning can be integrated into end-user modifiable systems. If users want to introduce new concepts to the system, they could show the system examples and counterexamples for this concept, and the system would learn these concepts by example (see Figure 6).

Other Possibilities for Integrating Adaptable and Adaptive Systems. Modification critics [Girgensohn 92] are adaptive system components that critique the adaptations of users. Adaptive components can be used to "suggest" to users how to adapt systems [Fi-

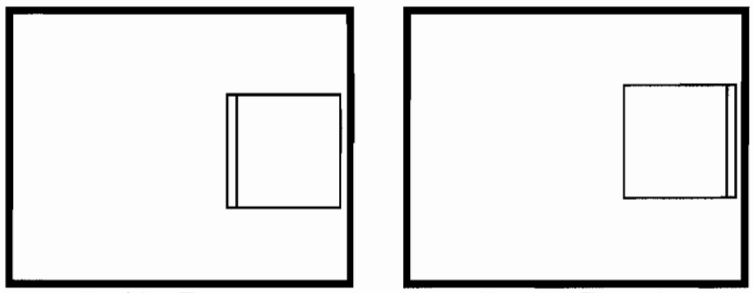


Figure 6: Learning a Rule with Examples

Our design environment for kitchen floorplans supports the creation of positive and negative examples, which can be used to integrate new knowledge into the system.

scher et al. 91a, Thomas & Krogsæter 93, Oppermann 92]. Adaptable systems could benefit from adaptive explanation components. It would also be beneficial if the interface for adaptations were adaptive to some extent so that it would be easier for users with different skills to do adaptations. In adaptive systems, it would be helpful if the users could adapt the parameters for the adaptation of the system, e.g., when to use which stereotype.

Other Efforts Integrating Adaptable and Adaptive Systems. InVision [Kass & Stadnyk 92] combines adaptable and adaptive mechanisms to address the information overload problem as it occurs in organizations (e.g., "who do I tell" and "who do I ask" type questions). InVision relies on explicitly represented models of the knowledge and information needs of members of the organization. It supports simultaneously the "computer as tool" paradigm (users explicitly build models using a specification by reformulation approach) and "computer as agent" paradigm (the system infers users' information needs based on observations of their interactions with data base systems) and it supports conflict resolution techniques for resolving contradicting information.

INFOSCOPE [Fischer & Stevens 91, Stevens 93] allows users to evolve the predefined system structure (for reading Usenet News) to suit their own semantic interpretations. Users can define virtual newsgroups. To do so, they can exploit information provided by agents who observe their behavior over periods of time and accumulate it in an $\mathbf{M}_2$-type model. INFOSCOPE illustrates how adaptive components can be used to drive the adaptation of a system.

FLEXCEL [Thomas & Krogsæter 93] is a research effort to add flexibility to a commercially available software systems (EXCEL). An empirical investigation of FLEXCEL [Oppermann 92] has shown that adaptive and adaptable systems are not alternatives, but are most promising when both features are linked to cooperate. Adding adaptable components to EXCEL demonstrated that such components (1) do not come for free (neither for designers nor for the users), (2) require users to shift from their tasks to a meta-task, and (3) will not be successful without extensive support mechanisms. The adaptive component of FLEXCEL assists users in the adaptation process by (1) preparing them to adapt the system, (2) presenting clues when to turn from the domain task to the meta-task of adaptation, and (3) achieving a balance between massive interruption and merely mute potential.

4 SHARED KNOWLEDGE SYSTEMS

Wittgenstein: "If a lion could speak would we understand her?"

Figure 7 illustrates communication breakdowns caused by a lack of shared understanding. The visitor cannot build up a coherent model. More of the information provided should

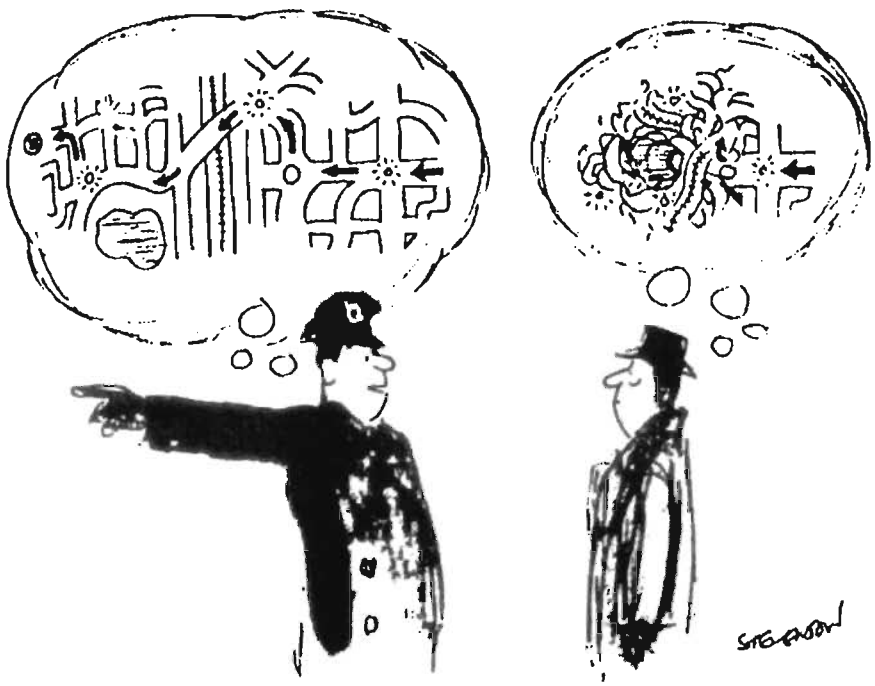


Figure 7: Lack of Shared Understanding

Drawing by Stevenson; © 1976 The New Yorker Magazine

have been put into the world (e.g., by drawing a map, thereby grounding the mutual understanding with a shared artifact). The structural model provided by the policeman is too detailed; it may be possible to avoid this by tailoring the explanations more to the goals and objectives of the visitor. The policeman could have reduced the complexity of his description using layers of abstractions or by providing minimalist explanations [Black et al. 87, Fischer et al. 90]. The policeman (by seeing the visitor the first time) is unable to model the background knowledge of the visitor, indicating that shared understanding is not a one-shot affair but a cooperative problem-solving effort [Moore 89, Fischer 90, Fischer & Reeves 92] requiring follow-up questions and detail-on-demand.

Adaptive and adaptable systems are desirable for the following reasons: (1) to support mutual intelligibility and reciprocal recognizability of our actions, enabled by common conventions for the expression of intent, and shared knowledge about typical situations, and (2) to support communicative economy (if the premise or rationale of an action can be assumed to be shared, it can be left unspoken). Design environments can be interpreted as shared knowledge systems [Resnick 91] at several levels:

- The domain-orientation of the design environments supports human problem-domain communication [Fischer & Lemke 88] by eliminating the need for users to deal with low-level computer-specific programming concepts, thereby allowing users to communicate directly with the problem domain rather than with the computer.
- The construction and the specification component allows users to articulate their specific problem-solving situation.
- Other system components (critics, catalog, argumentation, and simulation) illustrate the system's knowledge to users in the context of their task at hand.

Adaptive and Adaptable Components in the Context of Design Environments. Opponents of adaptive components based on $\mathbf{M_2}$-type models [Dumais 90, Hollan 90] have argued that there is little evidence from real systems that such models can be successfully constructed or exploited to enhance cooperative problem-solving activities. This criticism is justified by the lack of assessment studies analyzing the strength and weaknesses of systems' models of users as well as the absence of a true success story of such a system beyond research environments. These opponents are more in favor of adaptable systems and support mechanisms to enhance $\mathbf{M_1}$-type models. This view is based on the belief that (1) users know more about their interests, goals, and state of knowledge than what can be communicated, defined, abstracted, and exploited by most modeling mechanisms; and (2) knowledge is tacit, requiring the back-talk of the situation to trigger additional knowledge.

Design environments provide unique opportunities to enhance and integrate adaptive and adaptable components by exploiting shared knowledge structures:

- their domain orientation [Fischer 92] establishes a restricted set of objectives and goals, which users pursue in using them,

- the systems are used repeatedly over long periods of time by domain workers (making techniques such as "Edit Wear and Read Wear" [Hill et al. 92] an important information source to drive modifications),
- the design artifact is present in the design situation [Reeves & Shipman 92] (indicating some of the goals of users), and
- the specification component allows users to articulate the specifics of a design situation [Fischer & Nakakoji 91].

The shared knowledge is used (1) to make information more relevant to the task at hand (e.g., by prioritizing information structures in the palette, catalog, and argumentation), (2) to help users to create better artifacts, and (3) to support learning on demand [Fischer 91b].

5 CONCLUSIONS

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 [Suchman 87]. 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. Cooperative problem-solving systems need to take this asymmetry seriously and find alternative ways to enhance effective problem-solving activities rather than just relying on the simulation of human communication. Design environments offer interesting possibilities for integrating adaptive and adaptable components to increase the shared knowledge between users and computers.

REFERENCES

[Black et al. 87]
J.B. Black, J.M. Carroll, S.M. McGuigan, *What Kind of Minimal Instruction Manual Is The Most Effective*, Human Factors in Computing Systems and Graphics Interface, CHI+GI'87 Conference Proceedings (Toronto, Canada), ACM, New York, 1987, pp. 159-162.

[Carbonell 89]
J.G. Carbonell, *Introduction: Paradigms for Machine Learning*, Artificial Intelligence, Vol. 40, No. 1-3, 1989, pp. 1-9.

[Chin 89]
D.N. Chin, *KNOME: Modeling What the User Knows in UC*, in A. Kobsa, W. Wahlster (eds.), *User Models in Dialog Systems*, Springer-Verlag, New York, 1989, pp. 74-107.

[Clancey 86]
W.J. Clancey, *Qualitative Student Models*, Annual Review of Computing Science, Vol. 1, 1986, pp. 381-450.

[Draper 84]
S.W. Draper, *The Nature of Expertise in UNIX*, Proceedings of INTERACT'84, IFIP Conference on Human-Computer Interaction, Elsevier Science Publishers, Amsterdam, September 1984, pp. 182-186.

[Dumais 90]
S.T. Dumais, *Panel: User Modeling and User Interfaces*, Proceedings of AAAI-90, Eighth National Conference on Artificial Intelligence, AAAI Press/The MIT Press, Cambridge, MA, August 1990, pp. 1135-1136.

[Fain-Lehman & Carbonell 89]
J. Fain-Lehman, J.G. Carbonell, *Learning the User's Language: A Step Toward Automated Creation of User Models*, in A. Kobsa, W. Wahlster (eds.), *User Models in Dialog Systems*, Springer-Verlag, New York, 1989, pp. 163-194.

[Fischer 87]
G. Fischer, *A Critic for LISP*, Proceedings of the 10th International Joint Conference on Artificial Intelligence (Milan, Italy), J. McDermott (ed.), Morgan Kaufmann Publishers, Los Altos, CA, August 1987, pp. 177-184.

[Fischer 90]
G. Fischer, *Communications Requirements for Cooperative Problem Solving Systems*, The International Journal of Information Systems (Special Issue on Knowledge Engineering), Vol. 15, No. 1, 1990, pp. 21-36.

[Fischer 91a]
G. Fischer, *The Importance of Models in Making Complex Systems Comprehensible*, in D. Ackerman, M. Tauber (eds.), Mental Models and Human Computer Communication: Proceedings of the 8th Interdisciplinary Workshop on Informatics and Psychology (Schaerding, Austria), Elsevier Science, Amsterdam, 1991, pp. 3-36.

[Fischer 91b]
G. Fischer, *Supporting Learning on Demand with Design Environments*, Proceedings of the International Conference on the Learning Sciences 1991, Evanston, IL, August 1991, pp. 165-172.

[Fischer 92]
G. Fischer, *Domain-Oriented Design Environments*, Proceedings of the 7th Annual Knowledge-Based Software Engineering (KBSE-92) Conference (McLean, VA), IEEE Computer Society Press, Los Alamitos, CA, September 1992, pp. 204-213.

[Fischer et al. 85]
G. Fischer, A.C. Lemke, T. Schwab, *Knowledge-Based Help Systems, Human Factors in Computing Systems*, CHI'85 Conference Proceedings (San Francisco, CA), ACM, New York, April 1985, pp. 161-167.

[Fischer et al. 89]
G. Fischer, R. McCall, A. Morch, *JANUS: Integrating Hypertext with a Knowledge-Based Design Environment*, Proceedings of Hypertext'89 (Pittsburgh, PA), ACM, New York, November 1989, pp. 105-117.

[Fischer et al. 90]
G. Fischer, T. Mastaglio, B.N. Reeves, J. Rieman, *Minimalist Explanations in Knowledge-Based Systems*, Proceedings of the 23rd Hawaii International Conference on System Sciences, Vol III: Decision Support and Knowledge Based Systems Track, Jay F. Nunamaker, Jr (ed.), IEEE Computer Society, 1990, pp. 309-317.

[Fischer et al. 91a]
G. Fischer, A.C. Lemke, T. Mastaglio, A. Morch, *The Role of Critiquing in Cooperative Problem Solving*, ACM Transactions on Information Systems, Vol. 9, No. 2, 1991, pp. 123-151.

[Fischer et al. 91b]
G. Fischer, A.C. Lemke, R. McCall, A. Morch, *Making Argumentation Serve Design*, Human-Computer Interaction, Vol. 6, No. 3-4, 1991, pp. 393-419.

[Fischer et al. 92]
G. Fischer, J. Grudin, A.C. Lemke, R. McCall, J. Ostwald, B.N. Reeves, F. Shipman, *Supporting Indirect, Collaborative Design with Integrated Knowledge-Based Design Environments*, Human-Computer Interaction, Special Issue on Computer Supported Cooperative Work, Vol. 7, No. 3, 1992, pp. 281-314.

[Fischer et al. 93]
G. Fischer, K. Nakakoji, J. Ostwald, G. Stahl, T. Sumner, *Embedding Computer-Based Critics in the Contexts of Design*, Human Factors in Computing Systems, INTERCHI'93 Conference Proceedings, ACM, 1993, (in press).

[Fischer & Girgensohn 90]
G. Fischer, A. Girgensohn, *End-User Modifiability in Design Environments, Human Factors in Computing Systems*, CHI'90 Conference Proceedings (Seattle, WA), ACM, New York, April 1990, pp. 183-191.

[Fischer & Lemke 88]
G. Fischer, A.C. Lemke, *Construction Kits and Design Environments: Steps Toward Human Problem-Domain Communication*, Human-Computer Interaction, Vol. 3, No. 3, 1988, pp. 179-222.

[Fischer & Nakakoji 91]
G. Fischer, K. Nakakoji, *Making Design Objects Relevant to the Task at Hand*, Proceedings of AAAI-91, Ninth National Conference on Artificial Intelligence, AAAI Press/The MIT Press, Cambridge, MA, 1991, pp. 67-73.

[Fischer & Reeves 92]
G. Fischer, B.N. Reeves, *Beyond Intelligent Interfaces: Exploring, Analyzing and Creating Success Models of Cooperative Problem Solving*, Applied Intelligence, Special Issue Intelligent Interfaces, Vol. 1, 1992, pp. 311-332.

[Fischer & Stevens 91]
G. Fischer, C. Stevens, *Information Access in Complex, Poorly Structured Information Spaces*, Human Factors in Computing Systems, CHI'91 Conference Proceedings (New Orleans, LA), ACM, New York, 1991, pp. 63-70.

[Girgensohn 92]
A. Girgensohn, *End-User Modifiability in Knowledge-Based Design Environments*, Unpublished Ph.D. Dissertation, Department of Computer Science, University of Colorado, 1992, Also available as TechReport CU-CS-595-92.

[Henderson & Kyng 91]
A. Henderson, M. Kyng, *There's No Place Like Home: Continuing Design in Use*, in J. Greenbaum, M. Kyng (eds.), *Design at Work: Cooperative Design of Computer Systems*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1991, ch. 11, pp. 219-240.

[Hill 89]
W.C. Hill, *The Mind at AI: Horseless Carriage to Clock*, AI Magazine, Vol. 10, No. 2, Summer 1989, pp. 29-41.

[Hill et al. 92]
W.C. Hill, J.D. Hollan, D. Wroblewski, T. McCandless, *Edit Wear and Read Wear*, Human Factors in Computing Systems, CHI'92 Conference Proceedings (Monterey, CA), ACM, May 1992, pp. 3-9.

[Hollan 90]
J.D. Hollan, *User Models and User Interfaces: A Case for Domain Models, Task Models, and Tailorability*, Proceedings of AAAI-90, Eighth National Conference on Artificial Intelligence, AAAI Press/The MIT Press, Cambridge, MA, August 1990, pp. 1137.

[Kass & Finin 87]
R. Kass, T. Finin, *Modeling the User in Natural Language Systems*, Computational Linguistics, Special Issue on User Modeling, Vol. 14, No. 3, 1987, pp. 5-22.

[Kass & Stadnyk 92]
R. Kass, I. Stadnyk, *Using User Models to Improve Organizational Communication*, Proceedings of 3rd International Workshop on User Modeling (U M'92), The German Research Center for Artificial Intelligence, Dagstuhl, Germany, August 1992, pp. 135-147.

[Kobsa & Wahlster 89]
A. Kobsa, W. Wahlster (eds.), *User Models in Dialog Systems*, Springer-Verlag, New York, 1989.

[Lai & Malone 88]
K.-Y. Lai, T.W. Malone, *Object Lens: A "Spreadsheet" for Cooperative Work*, Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW'88), ACM, New York, September 1988, pp. 115-124.

[Lemke & Fischer 90]
A.C. Lemke, G. Fischer, *A Cooperative Problem Solving System for User Interface Design*, Proceedings of AAAI-90, Eighth National Conference on Artificial Intelligence, AAAI Press/The MIT Press, Cambridge, MA, August 1990, pp. 479-484.

[Mackay 92]
W.E. Mackay, *Co-adaptive Systems: Users as Innovators*, CHI'92 Basic Research Symposium, 1992.

[MacLean et al. 90]
A. MacLean, K. Carter, L. Lovstrand, T. Moran, *User-Tailorable Systems: Pressing the Issues with Buttons*, Human Factors in Computing Systems, CHI'90 Conference Proceedings (Seattle, WA), ACM, New York, April 1990, pp. 175-182.

[Moore 89]
J. Moore, *Responding to 'HUH': Answering Vaguely Articulated Follow-up Questions*, Human Factors in Computing Systems, CHI'89 Conference Proceedings (Austin, TX), ACM, New York, May 1989, pp. 91-96.

[Norman 82]
D.A. Norman, *Some Observations on Mental Models*, in D. Gentner, A.L. Stevens (eds.), *Mental Models*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1982, pp. 7-14.

[Norman 86]
D.A. Norman, *Cognitive Engineering*, in D.A. Norman, S.W. Draper (eds.), *User Centered System Design, New Perspectives on Human-Computer Interaction*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1986, pp. 31-62, ch. 3.

[Norman 93]
D.A. Norman, *Things That Make Us Smart*, Addison-Wesley Publishing Company, Reading, MA, 1993, Expected publication, early 1993.

[Oppermann 92]
R. Oppermann, *Adaptively Supported Adaptability*, Sixth European Conference on Cognitive Ergonomics, Human-Computer Interaction: Tasks and Organization (Balatonfuered, Hungary), September 1992, pp. 255-270.

[Polanyi 66]
M. Polanyi, *The Tacit Dimension*, Doubleday, Garden City, NY, 1966.

[Reeves & Shipman 92]
B.N. Reeves, F. Shipman, *Supporting Communication between Designers with Artifact-Centered Evolving Information Spaces*, Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW'92), ACM, New York, November 1992, pp. 394-401.

[Resnick 91]
L.B. Resnick, *Shared Cognition: Thinking as Social Practice*, in L.B. Resnick, J.M. Levine, S.D. Teasley (eds.), *Perspectives on Socially Shared Cognition*, American Psychological Association, Washington, D.C., 1991, pp. 1-20, ch. 1.

[Rich 83]
E. Rich, Users are Individuals: *Individualizing User Models*, International Journal of Man-Machine Studies, Vol. 18, 1983, pp. 199-214.

[Riesbeck & Schank 89]
C. Riesbeck, R.C. Schank, *Inside Case-Based Reasoning*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1989.

[Rittel 84]
H.W.J. Rittel, *Second-Generation Design Methods*, in N. Cross (ed.), *Developments in Design Methodology*, John Wiley & Sons, New York, 1984, pp. 317-327.

[Schoen 83]
D.A. Schoen, *The Reflective Practitioner: How Professionals Think in Action*, Basic Books, New York, 1983.

[Stallman 81]
R.M. Stallman, *EMACS, the Extensible, Customizable, Self-Documenting Display Editor*, ACM SIGOA Newsletter, Vol. 2, No. 1/2, 1981, pp. 147-156.

[Stefik 86]
M.J. Stefik, *The Next Knowledge Medium*, AI Magazine, Vol. 7, No. 1, Spring 1986, pp. 34-46.

[Stevens 93]
C. Stevens, *Helping Users Locate and Organize Information*, Unpublished Ph.D. Dissertation, Department of Computer Science, University of Colorado, 1993.

[Suchman 87]
L.A. Suchman, *Plans and Situated Actions*, Cambridge University Press, Cambridge, UK, 1987.

[Thomas & Krogsæter 93]
C.G. Thomas, M. Krogsæter, *An Adaptive Environment for the User Interface of Excel*, Proceedings of the 1993 International Workshop on Intelligent User Interfaces, Orlando, Fl, 1993, ACM Press.

[Trigg et al. 87]
R.H. Trigg, T.P. Moran, F.G. Halasz, *Adaptability and Tailorability in NoteCards*, Proceedings of INTERACT'87, 2nd IFIP Conference on Human-Computer Interaction (Stuttgart, FRG), H.-J. Bullinger, B. Shackel (eds.), North-Holland, Amsterdam, September 1987, pp. 723-728.