University of Colorado at Boulder

Center for LifeLong Learning and Design (L3D)

Department of Computer Science

ECOT 717 Engineering Center
Campus Box 430
Boulder, Colorado 80309-0430
(303) 492-1592, FAX: (303) 492-2844

Embedding Critics in Design Environments

Gerhard Fischer[1], Kumiyo Nakakoji[1,2], Jonathan Ostwald[1]
Gerry Stahl[1] and Tamara Sumner[1]

[1]Center for LifeLong Learning and Design (L3D)
Department of Computer Science and Institute of Cognitive Science
University of Colorado
Boulder, CO 80309-0430
gerhard@cs.colorado.edu

[2]Software Engineering Laboratory
Software Research Associates, Inc.
1-1-1 Hirakawa-cho, Chiyoda-ku, Tokyo 102, Japan

The Knowledge Engineering Review, Vol. 8:4, 1993, pp. 285-307

The Knowledge Engineering Review, Vol. 8:4, 1993, 285–307

Embedding critics in design environments

GERHARD FISCHER, KUMIYO NAKAKOJI,[1] JONATHAN OSTWALD,[2] GERRY STAHL[3] and TAMARA SUMNER

University of Colorado, Boulder, Colorado 80309-0430, USA (email: gerhard@cs.colorado.edu)

Abstract

Human understanding in design evolves through a process of critiquing existing knowledge and consequently expanding the store of design knowledge. Critiquing is a dialogue in which the interjection of a reasoned opinion about a product or action triggers further reflection on or changes to the artifact being designed. Our work has focused on applying this successful human critiquing paradigm to human-computer interaction. We argue that computer-based critiquing systems are most effective when they are embedded in domain-oriented design environments, which are knowledge-based computer systems that support designers in specifying a problem and constructing a solution. Embedded critics play a number of important roles in such design environments: (1) they increase the designer's understanding of design situations by pointing out problematic situations early in the design process; (2) they support the integration of problem framing and problem solving by providing a linkage between the design specification and the design construction; and (3) they help designers access relevant information in the large information spaces provided by the design environment. Three embedded critiquing mechanisms—generic, specific, and interpretive critics—are presented, and their complementary roles within the design environment architecture are described.

1 Introduction

Human understanding in design evolves through a process of critiquing (Fischer et al., 1991) existing knowledge and consequently expanding and refining the state of knowledge. Our work has focused on applying this human critiquing paradigm to human-computer interaction. Our experience with this approach is based on several years of system prototyping, the integration of cognitive and design theories, and empirical evaluation of these systems. Based on these experiences, we conclude that computational critiquing systems are most effective at supporting human designers when embedded in domain-oriented design environments (Fischer, 1992).

In section 2, we explain why the critiquing paradigm is essential for supporting the complex activity of design. Using illustrations from critiquing systems we have built, we demonstrate in section 3 how embedding in design environments enhances the computational critiquing process. Examples of our embedded critiquing system are drawn from HYDRA-KITCHEN, a residential kitchen design environment we have built. Section 4 explains three embedded critiquing mechanisms we have designed, implemented, and studied, called generic, specific and interpretive critics. Finally, in section 5 we assess some of the benefits of these embedded critiquing mechanisms.

[1] Also at: Software Engineering Laboratory, Software Research Associates, Inc., 1-1-1 Hirakawa-cho, Chiyoda-ku, Tokyo 102, Japan.
[2] Also at: Nynex Science and Technology Center, White Plains, New York, USA.
[3] Also at: School of Environmental Design, University of Colorado, Boulder, Colorado 80309, USA.

2 The critiquing approach

Critiquing is a dialogue in which the interjection of a reasoned opinion about a product or action triggers further reflection on or changes to the artifact being designed. For example, a kitchen designer might critique a kitchen floor plan in terms of building code violations, efficiency, safety concerns, or eventual resale value. An agent—human or machine—capable of critiquing in this sense is a *critic*. Computer-based critics are made up of sets of rules or procedures for evaluating different aspects of a product; sometimes each individual rule or procedure is referred to as a critic (Fischer et al., 1991).

2.1 Importance of human critiquing

Human critiquing plays an important role in design both in the growth of human knowledge and in terms of error elimination. By "human critiquing" we mean subjecting our designs and products to the scrutiny of other people, be they peers, domain specialists, or society in general.

Complex design activities prohibit an individual from knowing everything that is relevant; in addition, expertise is frequently controversial. Complex design situations can therefore be characterized by a "symmetry of ignorance" (Rittel, 1984), and the knowledge needed to solve a design problem is distributed among designers and their clients (Rittel & Webber, 1984). Critiquing is an important method for working within such a framework of distributed knowledge because it fosters a maximum of participation to activate as much of the distributed design knowledge as possible. In kitchen design, the designer and the homeowner take turns proposing ideas and criticizing each other's suggestions. In this way, the often tacit knowledge (Polanyi, 1966) that each party has can come into play and complement the other's partial grasp of the design problem.

Critiquing is ubiquitous. It is, for example, at the heart of the scientific method. Popper (1965) theorized that science advances through a cycle of conjectures and refutations. Scientists formulate hypotheses and put forth these conjectures for scrutiny and refutation by the scientific community. Besides contributing to the growth of knowledge, this critiquing cycle of conjectures and refutations is essential for creating a shared understanding within the scientific community and providing a stable base for future growth in scientific knowledge.

Critics play an important role in making designers aware of breakdown situations (Fischer, 1993). Petroski (1985) noted the importance of failure in the growth of engineering knowledge. For instance, when an airplane crashes, the Federal Aviation Adminstration sends a team of specialists to the site to determine the cause of the accident. In essence, these specialists are critiquing the plane's design and construction and current aviation practices. Over the years, this practice has contributed much to the growth of aviation knowledge in terms of both airplane design and improved safety regulations (Chambers & Nagel, 1985). In turn, this growth in knowledge contributes toward future error elimination; that is, planes with the same defect are repaired and aviation regulations are improved to prevent similar crashes.

The activity of critiquing plays an important role in engineering, science, and design in general. It produces many benefits, including the growth of knowledge, error elimination and the promotion of mutual understanding of all participants. Through the critiquing process, designers gain a better understanding of the design problem by hearing the different points of view of other design participants. In our work, we have taken this successful human critiquing paradigm and shown how it can be effectively applied to enhance human-computer interaction. In the remainder of this paper, the term "critiquing" will refer to computer-based critiquing systems.

2.2 Applying computer-based critiquing to design

Our design environments are *cooperative problem-solving systems* (Fischer, 1990) in which the computer system helps users design solutions themselves as opposed to having an expert system

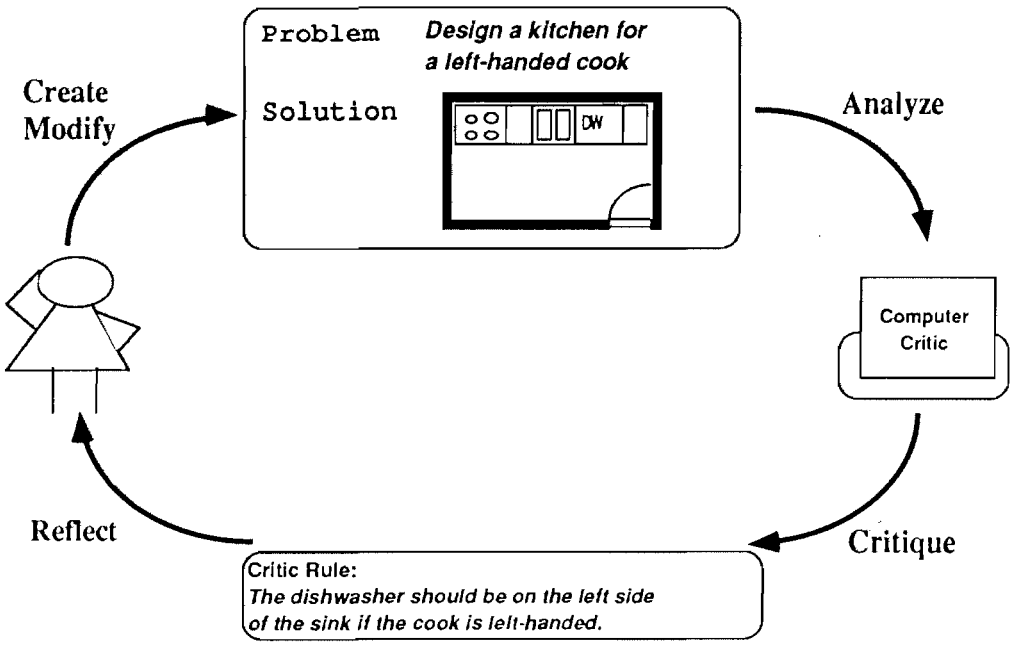

Figure 1 A cooperative problem-solving system has two agents—a human designer and a computer-based critic. Both agents contribute what they know about the domain to solving some problem. For the critiquing systems discussed in this paper, the human's primary role is to generate and modify solutions; the computer's role is to analyse these solutions and produce a critique for the human to consider in the next iteration of this process

design solutions for them. As illustrated in Figure 1, critiquing is integral to cooperative problem-solving systems. The core task of critics is to recognize and communicate debatable issues concerning a product. Critics point out problematic situations that might otherwise remain unnoticed. Many critics also advise users on how to improve the product and explain their reasoning. Critics thus help designers avoid problems and learn different views and opinions. Critiquing systems *augment* the ability of human designers to evaluate their solutions; decisions concerning whether or not to follow the critic suggestions are left up to the designers.

Critiquing systems are well suited for design tasks in complex problem domains in which the traditional expert systems or automated design approaches have proven inadequate. Such design tasks have the following characteristics: (a) knowledge about the design domain is *incomplete* and *evolving;* (b) the problem requirements can be specified only partially; and (c) necessary design knowledge is *distributed* among many design participants.

2.2.1 Knowledge about the design domain is incomplete and evolving

Some domains, such as user interface design (Lemke & Fischer, 1990) and lunar habitat design (Stahl, 1993), are not sufficiently understood; that is, creating a complete set of principles that exhaustively captures their domain knowledge is impossible. Complex problem domains are continually changing as new design knowledge is gained and old design knowledge becomes obsolete. For example, user interface design principles have certainly changed to accommodate the shift from primarily character-based user interfaces to sophisticated graphical user interfaces. Any system supporting design in complex domains must be able to evolve with the domain.

Expert systems and automated design approaches are infeasible in these complex situations in which all the potential relevant background knowledge cannot be articulated (Winograd & Flores, 1986). Because autonomous expert systems leave the human out of the decision process and all "intelligent" decisions are made by the computer, these systems require *a priori* a comprehensive knowledge base covering all aspects of the tasks being performed. Most expert systems also fail to adequately support the evolution of domain knowledge. First, expert systems typically do not

support the addition of knowledge by domain experts, and instead rely on knowledge engineers to acquire this knowledge from domain experts and subsequently codify it for the specific system. Second, expert systems have shown themselves to be brittle (Rittel & Webber, 1984); that is, a small shift in the problem domain can render an expert system's knowledge base obsolete and inoperative (Buchanan & Shortliffe, 1984).

An important aspect of embedded critiquing systems is their incremental nature; they do not need a large or comprehensive rule-base to be effective. Because critics are structured to be independent entities, adding or modifying a critic does not affect the behavior of the remaining critics. Parts of the critiquing system can remain operational and continue to support the design process while other parts undergo evolutionary change. In the HYDRA-KITCHEN system we have prototyped a "generic" critiquing mechanism that is knowledgeable about commonly accepted design principles and standard design practices. These principles are found in textbooks and training programs and are recognized by professional kitchen designers as being important aspects of producing a "good" floor plan. Although this general knowledge base is insufficient for automating the design of kitchen floor plans or for making a detailed analysis of the appropriateness of the design for a particular client, the generic critiquing system provides designers with valuable feedback concerning their floor plan designs. One study involving both amateur and expert kitchen designers showed that HYDRA's generic critics helped both categories of designers, even though its rule-base contained only 24 critic rules (Fischer et al., 1989).

2.2.2 The problem requirements can be specified only partially

Design problems are ill-defined: they cannot be precisely specified before attempting a solution (Rittel & Webber, 1984). Problem specifications reflect the designer's understanding of the problem framing and the problem solution. Researchers in situated cognition (Lave, 1988) and design (Schoen, 1983) have shown that designers arrive at solutions by iteratively reframing the problem—adjusting and refining their understanding of the problem framing and problem solution to reflect decisions made, means that may be chosen, materials available, and other changes in the context. Thus, problem specifications are not only incomplete, they are also dynamic in nature.

The expert system approach is based on the assumption that the problem to be solved can be fully articulated to the system *a priori*. The system can return a solution only if given a complete and accurate problem specification. Furthermore, changes in the problem specification can completely invalidate the expert system's proposed solution. Thus, expert systems are inadequate in ill-defined domains with partial and evolving problem specifications.

We have constructed a critiquing mechanism that supports design as a process of problem reframing. This "specific" critiquing mechanism enables only those critics pertinent to the current partial specification, and as such embodies domain knowledge concerning situation-specific design characteristics that not every design will share. In kitchen design, professional designers elicit this situation-specific knowledge from their customers using predefined questionnaires; the answers to these questionnaires form part of the kitchen specification. In HYDRA-KITCHEN, as the designer changes the problem specification, the "specific" critiquing mechanism brings different sets of critics to bear upon the design. This mechanism supports the coevolution of problem framing and problem solving by making explicit the relationship between the partial problem specification and the current design solution.

2.2.3 Necessary design knowledge is distributed among many design participants

Design domains such as network design are so large and complicated and have so many subdomains that no single person can know all there is to know (Fischer, 1991). In such complex domains, the necessary design knowledge is distributed among many participants and most design work is done by teams whose members have different areas of expertise (Hackman & Kaplan, 1974; Johansen, 1988). When designing in ill-defined domains, there are no "optimal" solutions (Simon, 1981). Conflicts in opinion about how to proceed often arise due to differences in the designers' areas of

expertise, their personal styles, and their particular problem framing. Often, such conflicts are resolved and design proceeds after designers present reasoned arguments supporting their opinions for discussion and negotiation.

Our critiquing systems support design as a deliberative and interpretative process. Critiquing systems contain a collection of critics that embody different areas of domain expertise, different design styles, and often diverging opinions. Our "interpretive" critiquing mechanism supports designers with varying interests and differing areas of expertise to work together by allowing design knowledge to be defined and bundled into personal or topical groupings. Using this mechanism, designers can examine their design from many different perspectives in which each perspective brings different design knowledge and critics to bear upon the current design.

All of our critiquing mechanisms—generic, specific and interpretive—support design as a deliberative process. Besides simply pointing out a potential flaw in the design, these critics offer a reasoned opinion as to why their suggestion should or should not be followed. This interaction style typifies cooperative problem-solving systems: it is the role of the critiquing system to bring relevant design knowledge to the designer's attention; it is the role of the designer to evaluate the trade-offs and make the final decisions.

3 Embedding critics in integrated design environments

Our early research focused on building and evaluating general purpose (i.e., not domain-oriented) critiquing mechanisms (Fischer et al., 1991). During later work, we became interested in building domain-oriented design environments (Fischer, 1992). In the last few years, we have merged these two research interests by embedding critiquing mechanisms into domain-oriented design environments. This embedding enhances both the richness of the critiquing process and the ability of our design environments to support the complex activity of design. This section discusses early critiquing systems we have built and how they contributed to the development of the multifaceted architecture, HYDRA, for design environments. A scenario using HYDRA-KITCHEN illustrates how the embedded critiquing mechanisms integrate the various components in the design environment.

3.1 Analyses of early critiquing systems

Critical analyses of our early stand-alone critiquing systems (Fischer et al., 1991) and systems built by others (Burton & Brown, 1982; Silverman, 1992), combined with empirical evaluations, led us to realize that the challenge in building critiquing systems is not simply to provide feedback: the challenge is to say *the right thing at the right time*. Our analyses identified several shortcomings in early critiquing systems that hindered their ability to say the "right" thing at the "right" time:

- lack of domain orientation;
- insufficient facilities for justifying critic suggestions;
- lack of an explicit representation of the user's goals;
- no support for different individual perspectives;
- timing problems with critic intervention strategies.

3.1.1 Lack of domain orientation

LISP-CRITIC (Fischer, 1987) allows programmers to request suggestions on how to improve their code. The system proposes transformations that make the code more cognitively efficient (i.e., easier to read and maintain) or more machine efficient (i.e., faster or smaller). However, the lack of domain orientation limits the depth of critical analysis the critiquing system can provide. Without domain knowledge, critic rules cannot be tied to higher level concepts; LISP-CRITIC can answer questions such as whether the Lisp code can be written more efficiently, but it cannot assist a user in deciding whether the code can solve a specific problem.

3.1.2 Insufficient facility for justifying critic suggestions

FRAMER (Lemke & Fischer 1990) enables designers to develop window-based user interfaces on Symbolics Lisp machines. FRAMER's knowledge base contains design rules for evaluating the completeness and syntactic correctness of the design as well as its consistency with interface style guidelines. Evaluations of FRAMER showed (1) that many users did not understand the consequences of following the critic's advice or why the advice was beneficial to solving their problem, and (2) that when users do not understand why a suggestion is made, they tend to blindly follow the critic's advice whether or not it is appropriate to their situation. FRAMER provided short explanations to address this problem. However, in design there are not always simple answers; access to argumentative discussions detailing the pros and cons of a particular suggestion are necessary (Rittel & Webber, 1984).

3.1.3 Lack of an explicit representation of the user's goals

JANUS (Fischer et al., 1989) is a step toward addressing the previous shortcomings. JANUS allows designers to construct kitchen architectural floor plans. It contains two integrated subsystems: a domain-oriented kitchen construction kit and an issue-based hypermedia system containing design rationale. Critics respond to problems in the construction situation by displaying a message and providing access to appropriate rationale in the hypermedia system. However, these critics often give spurious or irrelevant advice resulting from the lack of an explicit representation of the user's task. The only task goal built into JANUS is one of building a "good" kitchen; that is, a kitchen that conforms to commonly accepted standards and design practices. With an explicit model of the designer's intentions for a *particular* design, critics can be selectively enabled based on this model and provide less intrusive and more relevant advice.

3.1.4 No support for different individual perspectives

It is not possible to anticipate all the knowledge necessary for a critiquing system to say the "right" thing in every design situation. Design domains are continually evolving as new knowledge is gained. JANUS-MODIFIER (Fischer & Girgensohn, 1990) was developed to respond to this problem by making the domain knowledge (including critics) end-user modifiable. But being able to add new knowledge is not sufficient; different users must be able to organize and manage design knowledge and critics to reflect *their* perspectives on design. Design environments need to support interpretation of a problem from many perspectives (technical, structural, functional, aesthetic, personal), and critique accordingly.

3.1.5 Timing problems with critic intervention strategies

A number of systems (Fischer et al., 1985; Burton & Brown, 1982) investigated critic intervention strategies, which determine when and how a critic should signal a potential problem. This research focused on studying *active* versus *passive* intervention strategies. Active critics continually monitor user actions and make suggestions as soon as a problematic situation is detected. Passive critics are explicitly invoked by users to evaluate their partial design.

A protocol analysis study (Lemke & Fischer, 1990) showed that passive critics were often not activated early enough in the design process to prevent designers from pursuing solutions known to be suboptimal. Often, subjects invoked the passive critiquing system only after they thought they had completed the design. By this time, the effort of repairing the situation was expensive. In a subsequent study using the same design environment, an active critiquing strategy was shown to be more effective by detecting problematic situations early in the design process.

However, our interactions with professional designers showed that active critics are not a perfect solution either: they can disrupt the designer's concentration on the task at the wrong time and interfere with creative processes. Interruption becomes even more intrusive if the critics signal breakdowns at a different level of abstraction compared to the level of the task users are currently engaged. For example, if the designer is currently concerned about where the refrigerator should

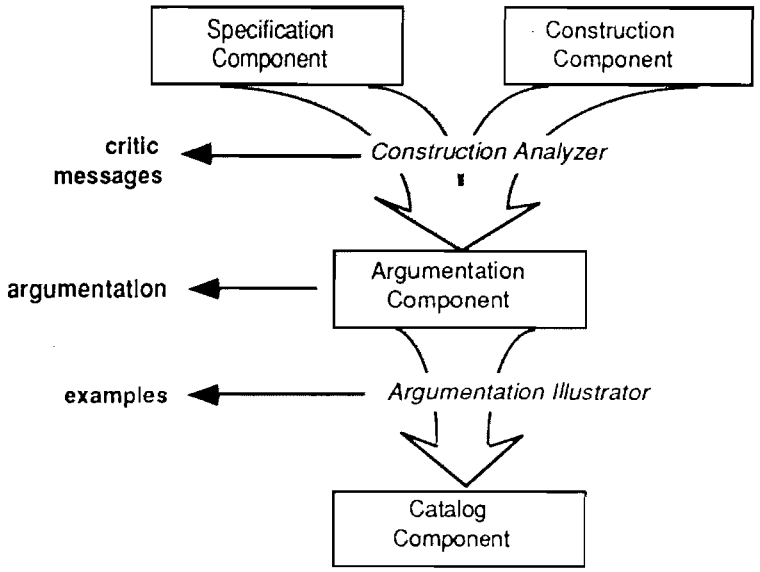

Figure 2 The critiquing process with HYDRA. The links between the components—the *Construction analyser* and the *Argumentation illustrator*—are crucial for exploiting the synergy of the integration.

be located in a kitchen floor plan, then a critic suggestion that a double-bowl sink is better than a single-bowl sink is probably inappropriate and distracting at this point in time.

What is needed is a critiquing system that: (1) alerts designers to problematic solutions; (2) avoids unnecessary disruptions; and (3) allows users to control the critic's intervention strategy. Embedding critics in design environments allows users to *control* critic intervention through interaction with the construction, specification, and perspective design components built into the design environment.

3.2 HYDRA: *A multifaceted architecture for design environments*

Design environments are computer programs that support designers in concurrently specifying a problem and constructing a solution. Design environments provide information repositories to store domain knowledge and allow designers to accumulate additional domain-knowledge through interaction with the environment.

HYDRA (Figure 2 represents its components schematically; Figure 3 provides a screen image) contains design creation tools in the form of a construction component and a specification component. Design information repositories are provided in the form of argumentation and catalog knowledge bases. The architecture is *multifaceted* because these components provide multiple representations of both the current design and underlying domain knowledge. The critiquing mechanisms integrate these facets in the design environment architecture. The various representations are managed by the following four components:

- The *construction component* is the principal medium for modelling a design. It provides a palette of domain-oriented design units, which can be arranged in a work area using direct manipulation. Design units represent primitive elements in the construction of a design, such as sinks and stoves in the domain of kitchen design. Critics can be tied to these domain-oriented design units and to relationships between design units.
- The *specification component* allows designers to describe abstract characteristics of the design they have in mind. The specifications are expected to be modified and augmented during the design process, rather than to be fully articulated at the beginning. The specification provides the system with an explicit representation of the user's goals. This information can be used to

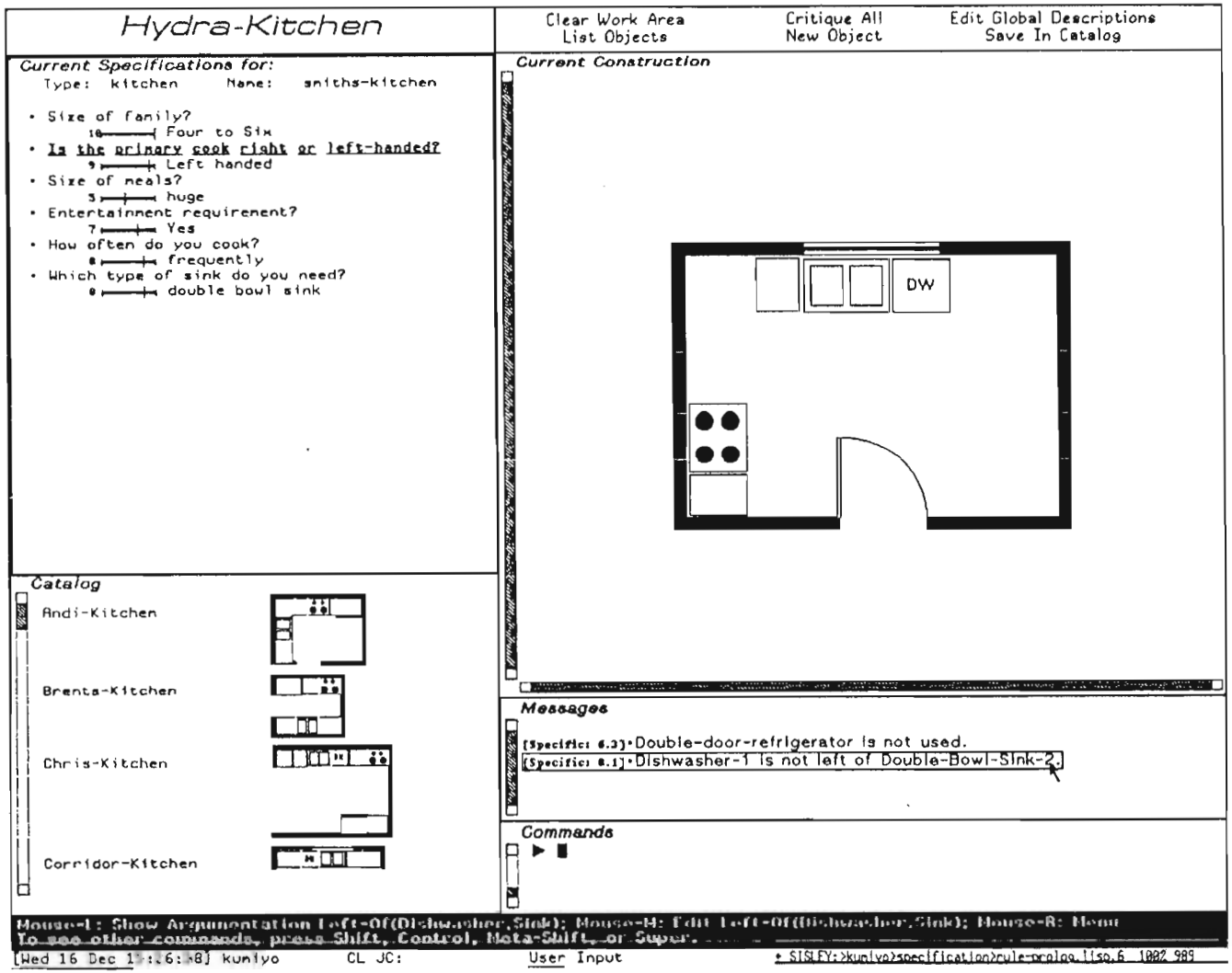

Figure 3 Screen image of HYDRA-KITCHEN. The "Current Specification" window shows a summary of currently selected answers using the specification component. An indicator attached to each of the selected answers allows users to assign weights of importance to the specified item in order to set priorities. The "Catalog" window shows previous kitchen designs that can be examined or reused. The "Current Construction" window shows a partial construction being built using components provided in a palette of kitchen design units (not shown). The "Messages" window is used to present critic notification messages. The number attached to the critic message is a weighted measure indicating the relevance of the fired critic.

tailor both the critic suggestions put forth and the accompanying explanations to the user's task at hand.

- The *argumentative hypermedia component* contains design rationale based on the procedural hierarchy of issues (PHI) structure (see Figure 5) (McCall, 1987; Conklin & Begeman, 1988). The PHI structure consists of issues, answers, and arguments about decisions made during the course of design. Users can annotate and add argumentation as it emerges during the design process. Argumentation is a valuable component in a critic's explanation; it identifies the pros and cons of following a critic suggestion and helps the user to understand the consequences of following a suggestion.
- The *catalog component* provides a collection of previously constructed designs. These illustrate examples within the space of possible designs in the domain and support reuse (Prieto-Diaz & Freeman, 1987) and case-based reasoning (Kolodner, 1991). Catalog entries are also important components in a critic's explanation. Often, a critic does not suggest a course of action but instead points out a deficiency in the current design; catalog entries can then be used as specific examples illustrating sample solutions that address a deficiency noted by a critic.

This architecture derives its power from the *integration* of its components. When used in combination, each component augments the value of the others in a synergistic manner. The components of the architecture are integrated by two linking mechanisms (see Figure 2). Together, these linking mechanisms support the critiquing process by providing critic messages, explanatory argumentation, and illustrative examples:

- The *construction analyser* is the core critiquing component in HYDRA. This mechanism analyses the design construction for compliance with the currently enabled set of critic rules. When a lack of compliance is detected, the critic signals a breakdown and provides entry into the exact place in the argumentative hypermedia component in which the appropriate explanation is located.
- The *argumentation illustrator* can retrieve both positive and negative catalog examples to illustrate the problematic situation detected by the *construction analyser*. Providing specific examples is essential, because the explanation given in the form of argumentation is often highly abstract and conceptual. Concrete design examples that match this explanation assist designers in understanding the potential problem, assessing the design situation, and devising a solution.

In addition to the construction and argumentation components of its predecessor JANUS, HYDRA supports a specification component (Fischer & Nakakoji, 1991) and a catalog of designs. The specification format is based on questionnaires used by professional kitchen designers to elicit their customers' requirements, such as the kitchen owner's cooking habits and family size. Each component in HYDRA contains design knowledge that can be used by an embedded critiquing mechanism to overcome the deficiencies of the stand-alone systems previously described.

As mentioned in section 2.2, we have studied three classes of embedded critiquing mechanisms: generic, specific and interpretive critics. These mechanisms embody different types of design knowledge, and correspond to three dimensions of embedding. *Generic critics* are embedded in the construction and use domain knowledge concerning desirable spatial relationships between design units to detect problematic situations in the partial design construction. *Specific critics* are embedded in the partial specification and take advantage of additional knowledge in the partial specification to detect inconsistencies between the design construction and the design specification. *Interpretive critics* are embedded in a perspective mechanism that enables designers to create topical groupings of critics and design knowledge; such groupings support designers in examining their artifacts from different viewpoints. The argumentation and catalog components provide rich sources of domain knowledge that all three mechanisms use in their explanation process when communicating with the designer.

The following section provides a scenario depicting how kitchen designers work within the HYDRA environment. The scenario describes the three critiquing mechanisms, and it illustrates the benefits derived from embedding these mechanisms in the multifaceted architecture.

3.3 Scenario illustrating generic, specific and interpretive critics

Imagine that Bob, a professional kitchen designer, has been asked to design a kitchen for the Smith family. The partial specification of the Smith's kitchen is articulated using HYDRA, as shown in Figure 3.

Bob begins working on a floor plan in the construction area. He moves the dishwasher next to the cabinet. Bob's action triggers a *generic critic*, and the message "The dishwasher is too far from the sink" is displayed. Generic critics reflect knowledge that applies to all designs, such as accepted standards, building codes, and domain knowledge based on physical principles. Often, this generic knowledge can be found in textbooks, training curricula, or by interviewing domain practitioners. Bob highlights the critic's message and elects to see its associated argumentation. The argumentation explains that plumbing guidelines require the dishwasher to be within one meter of the sink. Bob follows the critic's suggestion and moves the dishwasher next to the right side of the sink (for details, see Fischer et al., 1991).

This action triggers a *specific critic* with the rule "If you are left-handed, the dishwasher should be on the left side of the sink". Specific critics reflect design knowledge that is tied to situation-specific physical characteristics and domain-specific concepts that not every design will share. These critics are constructed dynamically from the partial specification to reflect current design goals. This particular critic rule was activated because Bob specified that the primary cook is left-handed (see Figure 3). Bob examines the supporting argumentation "Having the dishwasher to the left of the sink creates an efficient work flow for a left-handed person". Bob decides this is an important concern and puts the dishwasher on the left side of the sink.

Then Bob remembers that the Smiths are remodelling mainly to increase their property value in anticipation of selling in two years. So Bob decides to examine his design from a resale-value perspective. When Bob switches to the resale-value perspective, an *interpretive critic* is triggered with the rule "The dishwasher should be on the right side of the sink". Interpretive critics support design as an interpretive process by allowing designers to interpret the design situation from different perspectives according to their interests. In this perspective, the critic about the dishwasher and sink has been redefined and its associated rationale has been modified. Now the argumentation says "Optimizing your kitchen for left-handed cooks can adversely affect the house's resale value since most kitchen users are right-handed". Bob decides that enhancing the Smiths' resale value is the more important consideration and moves the dishwasher. As long as he remains in the resale-value perspective, Bob will be informed by the critics whenever they detect a feature negatively affecting resale value. Additionally, the critics will provide Bob access to argumentation concerning designing for resale.

4 Three embedded critiquing mechanisms

This section describes in detail three embedded critiquing mechanisms—*generic, specific* and *interpretive*. Examples of how these three critic styles are deployed were illustrated in the previous scenario. In all three mechanisms, critic knowledge is captured by rules with condition and action parts. The *condition* clause checks whether a certain situation exists in the current design construction. The *action* clause notifies the designer that a particular situation has been detected. Figure 4 illustrates a condition-action critic rule in which the condition checks if the stove is away from the window; the action part notifies the designer that "the stove is not away from the window".

For all three mechanisms, the basic critiquing process consists of the following phases: (1) the set of appropriate critic rules to be enabled is identified; (2) the design construction is then analysed for compliance with the currently enabled set of critic rules; (3) when a lack of compliance is detected, the critic signals a possible problem and provides entry into the argumentative hypermedia component in which the appropriate explanation is located; and (4) concrete catalogue examples that illustrate the explanation given in the form of argumentation can

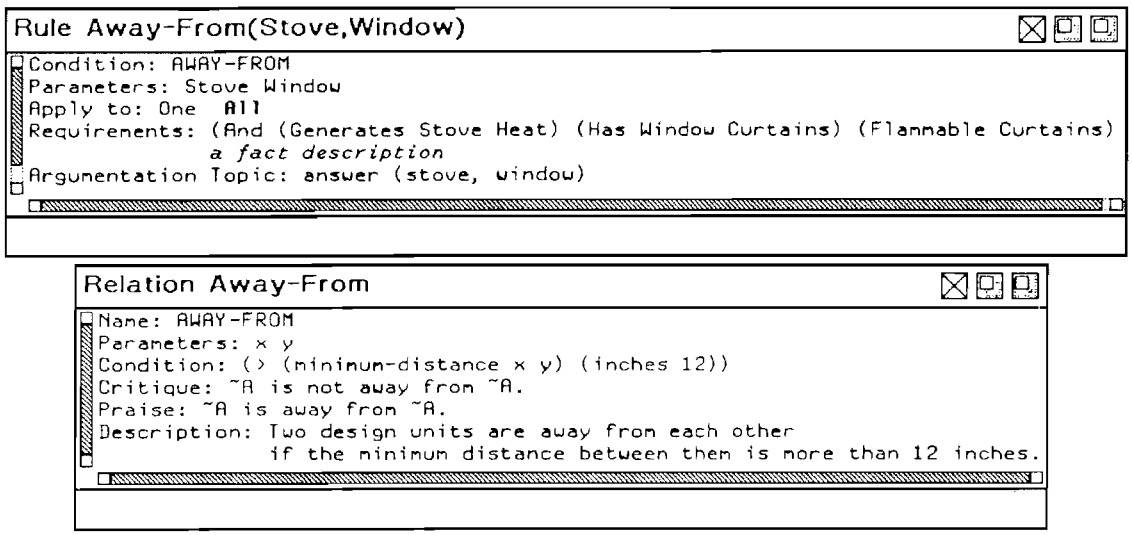

Figure 4 The "stove should be away from the window" critic rule and the definition of the "away-from" spatial relation

Table 1 The critic mechanisms—generic, specific and interpretive—differ in how they enable critic rules, the rules' scope of applicability, and the types of design knowledge each mechanism is best suited to represent

	How enabled	*Applicability*	*Design knowledge*	*Example*
Generic	Enabled by placing design units into the construction area	All designs	Standards	Cabinets should be 150 cm above floor
			Physical principles	Heat ignites flammable objects
Specific	Enabled by the partial specification	Specific design	Situation characteristics	Cook is left-handed and 150 cm in height
			Abstract domain concepts	Efficiency; safety
Interpretive	Enabled by the currently active design perspective	Specific perspective	Multiple interpretations of domain concepts	Cabinet height: convenient for cook
				Cabinet height: desirable for resale value

optionally be delivered (Fischer et al., 1991). As illustrated in Table 1, the three critic mechanisms differ mainly in terms of how they enable critic rules and in the types of design knowledge embodied in their rules.

- *Generic critics* (Fischer et al., 1991) are enabled by the placement of design units into the construction area. These critics apply to all designs containing the design unit to which the critics are attached. Generic critics reflect knowledge that is applicable to all designs, such as accepted standards or regulations or domain knowledge based on physical principles (see Table 1).
- *Specific critics* (Nakakoji, 1993) are constructed dynamically to reflect the designer's goals as they are stated explicitly in the specification component. These critics apply only to the design situation currently under construction. Specific critics reflect design knowledge that is tied to situation-specific physical characteristics and domain-specific concepts that not every design will share.
- *Interpretive critics* (Stahl, 1993) provide a mechanism for supporting design as an interpretive process; that is, they are a response to the recognition that domain concepts such as "cabinet height" and "efficiency" can have more than one definition or interpretation depending upon the

current situation and the designer. Interpretive critics allow designers to view their work from multiple perspectives by creating, managing and selectively activating different sets of design knowledge.

Specific examples illustrating each of these critic mechanisms will be discussed below. Generic critics will be used to discuss the basic critiquing process described at the beginning of this section. The three mechanisms for embedded critics differ from one another primarily in how they determine which set of critic rules should be enabled. The discussion of specific critics and interpretive critics will focus on how these mechanisms determine which critics are currently enabled.

4.1 Generic critics

Generic critics reflect knowledge that applies to all designs such as accepted standards, building codes and domain knowledge based on physical principles. Often, this generic knowledge can be found in textbooks, training curricula, or by interviewing domain practitioners. A generic critic representing an accepted kitchen design standard is the cabinet height critic. Kitchen designers agree that unless more specific information regarding the primary cook is known, the top cabinets should be placed 150 cm above the floor. A generic critic reflecting domain knowledge based on safety principles is the "stove should be away from the window" rule shown in Figure 4. This rule reflects the principle that objects that generate heat (e.g., the stove) should not be placed under flammable objects (e.g., the curtains on the window).

Generic critics in HYDRA are implemented as object-oriented methods of appliances and other design units in the design construction. When the design construction is altered, all design units implicated by the changes evaluate their critic methods. These methods are defined and parameterized by the information in property sheets such as those shown in Figure 4. For example, the rule box shown defines a generic critic for stoves. This method checks that the stove is "away from" all windows in the construction area.

The condition away-from is defined in the relation property sheet as taking two objects and evaluating whether or not the minimum distance between them is greater than 12 inches. The corresponding message for display if this condition is not met is the critique: the first object "is not away from" the second object.

The critic defined in the rule sheet applies this relation to the stove as the first parameter and sequentially to each window in the construction as the second parameter. The definition specifies that this rule shall be applied to windows (Apply to: All) because stoves should be away from all windows to prevent fires. Other critic rules specify only that there should exist at least one object in the construction (Apply to: One) that matches the condition relation with the first parameter—for example, the dishwasher should be near at least one sink.

Further requirements can be specified for the applicability of the critic rule. These applicability requirements make use of domain concepts like "generates heat", "has curtains" and "is flammable". In the example rule, a stove has to be away from a window only if the stove generates heat (e.g., it is not a microwave), if the window has curtains, and if the curtains are flammable. Finally, the definition of the critic lists a topic in the argumentation issue-based that will be displayed if this critic fires and the user selects the critic message.

All generic critics in HYDRA are defined through property sheets like these for rules and relations. Using these property sheets, designers are able to modify the definitions of existing critics and to create additional critics.

Critics inform designers of potentially problematic situations by using a three-tiered approach that involves simple notification, supporting argumentation and specific examples. First, the critic signals the designer of a potentially problematic situation with a simple initial notification message. The form of this initial notification message is defined by the critique phrase in the spatial relation definition. The critic shown in Figure 4 would display the message "Stove-1 is not away from

Window-1". Variables in the notification string are resolved into specific design units by the critic rule using the spatial relation. Associating notification messages with the spatial relations allows these messages to be shared by many critic rules. The downside of this approach is that the notification message signals only that a spatial relation was detected and does not report why this is significant.

As discussed in section 3.1, our work has shown that such "one-shot" notifications, which merely identify a situation, are inadequate. Critics that support design as an argumentative process (Rittel & Webber, 1984) should be capable of presenting different alternatives and opinions and each alternative's corresponding advantages and disadvantages. The critiquing systems use the argumentation component of HYDRA to provide the second tier of explanation, thereby "making argumentation serve design" (Fischer et al., 1991).

Each critic rule has an associated link into the argumentation component where issues pertaining to the situation identified by the critic are discussed. For the critic in Figure 4, the associated link is found in the slot "Argumentation Topic: answer (stove, window)". The designer can view the critic's associated design rationale by selecting the initial notification message displayed in the Message area (Figure 3). Because design rationale contains design issues accompanied by positive and negative argumentation, critic explanations in this form help the designer understand why the current design situation may be significant or problematic.

Sometimes designers may not understand the arguments made in the design rationale or they may understand the arguments but not know what action to take. In these situations, providing designers with specific examples can be helpful. The third tier of critic explanation delivers specific examples upon request that illustrate the issue being discussed. Designers can select an issue in the argumentation and request to see a positive example or a counter example. As illustrated in Figure 4, critic conditions are associated with argumentation issues. When the designer requests to see an example of a specific issue, the *argumentation illustrator* (see Figure 5) takes the critic condition associated with the selected argumentation issue and searches the catalog component for examples that fulfill the condition.

4.2 Specific critics

In HYDRA specification knowledge is related to: (1) situation-specific physical characteristics such as the size and shape of the kitchen or the owner's height; (2) specified requirements such as "a dishwasher should be included"; and (3) abstract domain concepts such as safety and efficiency. The specification issues were derived from questionnaires used by professional kitchen designers (Nakakoji, 1993).

Specific critics evaluate the construction situation for compliance with the partial specification. They reduce the intrusiveness of a critiquing system by narrowing the enabled critics to those that are relevant to the task at hand as determined from the partial specification. Specification-linking rules (Fischer & Nakakoji, 1991) are used to dynamically identify the set of specific critics to be enabled.

The specification consists of issue/answer pairs (see Figures 3 and 6). A specification linking rule represents a dependency between an issue/answer pair in the specification and associated pro and con arguments in the argumentation component. As shown in Figure 6, a specification linking rule connects the argumentation issue "Where should the stove be located?" with the specification item "Is safety important to you?" The shared domain distinction "safety" is used to establish a dependency between this particular specification item and the argumentation issue.

A critic condition is associated with each answer in the specification, and a domain distinction is associated with each argument. Domain distinctions are a vocabulary for expressing domain concepts such as safety or efficiency. Whenever the designer modifies the specification, the critiquing system recompiles the specification-linking rules to reflect the newly relevant domain distinctions. In this way, critiquing criteria are tied to a representation of the partially articulated goals of a specific design project.

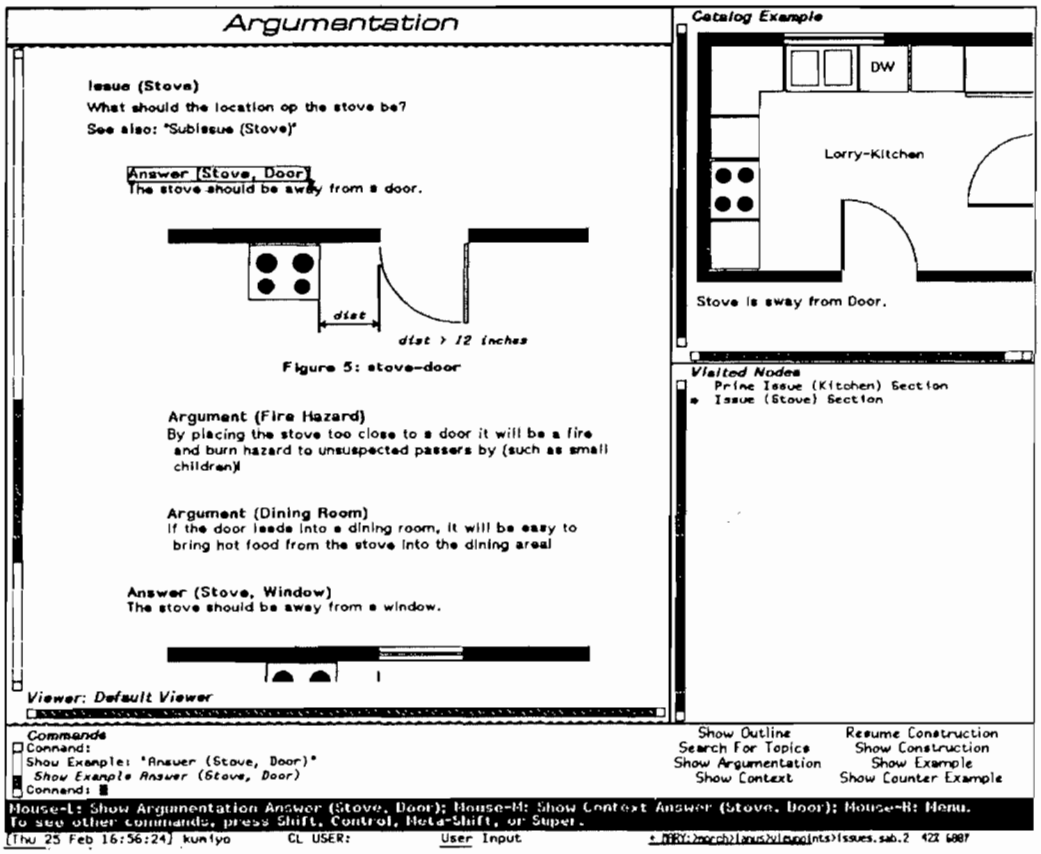

Figure 5 Argumentation consists of issues, answers and arguments supporting or refuting answers. The designer can view the stove-away-from-window critic's associated design rationale by selecting the initial notification message displayed in the Message area (e.g. "Stove-1 is not away from Window-1") of Figure 3. The arguments shown explain why many kitchen designers believe windows and stoves should not be adjacent. Choosing the menu item "Show Example" causes example designs that illustrate the answer advocated in the argumentation to be delivered to the designer.

The operation of the specification-linking rules can best be conveyed with an example. Assume the designer knows that the kitchen owners have young children and he specifies that having a safe (child-proof) kitchen is very important (Figure 6). The domain distinction associated with this specification item is "safety". In the argumentation, answers (e.g., "the stove should be away from all doors") are associated with critic conditions (e.g., "away-from stove door"). Pro and con arguments are associated with domain distinctions. In Figure 6, the domain distinction "safety" is associated with the pro argument and the domain distinction "efficiency" is associated with the con argument.

Specification-linking rules link the domain distinctions activated in the specification with the appropriate critic condition. First, the argumentation is analysed until the domain distinction activated in the specification (safety) is found. If the domain distinction is associated with a pro argument, then a specification-linking rule is created with the form: domain distinction implies critic condition. If the domain distinction is associated with a con argument, then a specification-linking rule is created with the form: domain distinction implies not critic condition. The specification-linking rules "safety implies stove away-from door" and "efficiency implies stove not away-from door" can be derived from the example in Figure 6. Whenever the designer modifies the specification, the critiquing system recomputes the specification-linking rules. For the partial specification shown in Figure 6, specification-linking rules supporting the notion of safety will be constructed. The right side of the specification rules are the enabled critic conditions used to evaluate the design construction for adherence to the current specification.

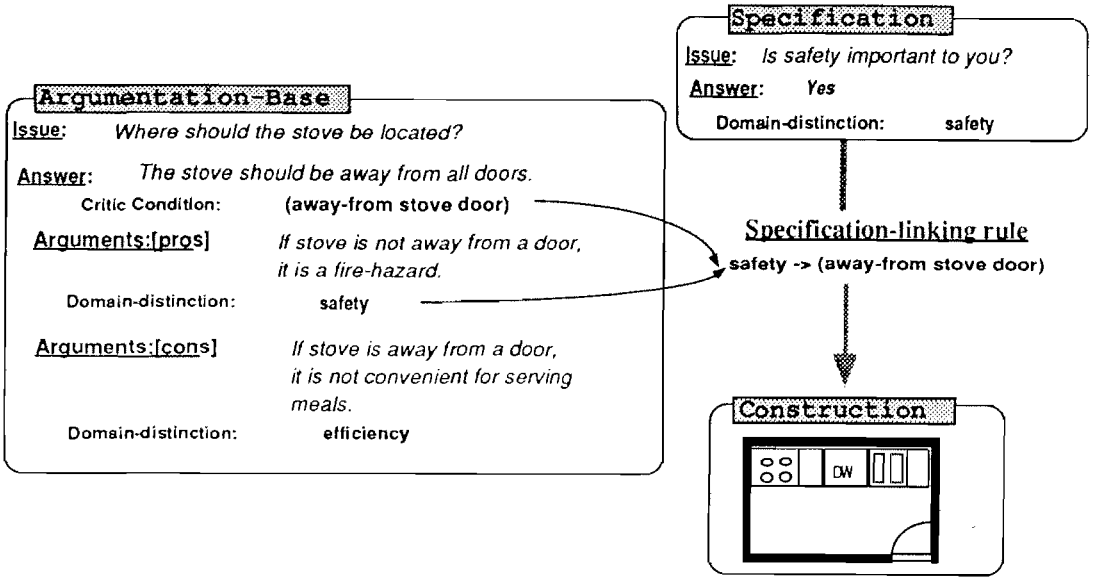

Figure 6 Derivation of the Specification-Linking rules. The domain distinction associated with a specification item is paired with a matching pro or con argument in the hypermedia issue base. The critic condition associated with an answer is linked with the domain distinction to form a specific critic rule.

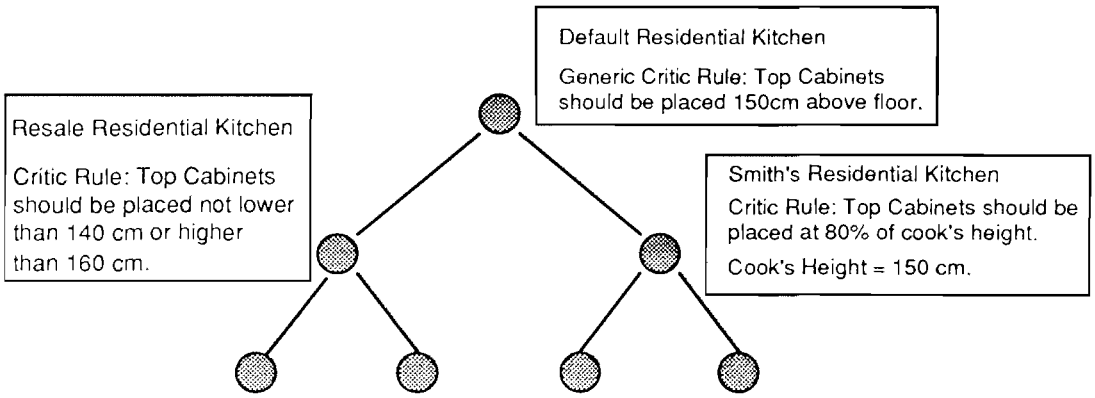

Figure 7 Perspectives are arranged in an inheritance network. Three perspectives—a "default kitchen", "Smith's kitchen" and a "resale kitchen"—are shown. The preferred placement of the top cabinets depends on the perspective selected. The critic rule analysing the placement of the top cabinets is redefined within each of the three perspectives.

Often, conflicts between specific critics arise. The designer could have specified that he was concerned with both safety and efficiency. For example, having the stove to the left of the refrigerator may be efficient, but it may also be less safe if this places the stove next to the door. Using the specification component, the designer cannot only state which concepts are of interest, he can also articulate his level of interest by weighting specification items. The critiquing system uses these weights to help prioritize critic activity. When a critic fires, it displays an importance weight next to the initial notification message that reflects the weights assigned to the specification items that enabled the particular critic rule (see Figure 3). The designer can then take these relative weights into account when deciding to respond to the critic messages.

4.3 Interpretive critics

Design can be viewed as an interpretive process (Stahl, 1993). Designers and their clients interpret the design situation according to personal backgrounds, experiences, and concerns. This means

that there cannot be a unique set of domain knowledge that is adequate for all people and all interests. We have prototyped a design environment (Stahl, 1992) with *perspectives* (Bobrow & Goldstein, 1980) to provide alternative views or approaches to given design situations. The perspectives mechanism organizes all the design knowledge in the system. It allows items of knowledge to be bundled into personal or topical groupings or versions. For instance, a resale-value perspective might include critics and design rationale pertinent to homeowners concerned about their home's resale appeal. A kitchen design environment might have perspectives for evaluating kitchens from the perspective of an electrician, a plumber, an interior designer, a realtor, a mortgage writer or a city inspector. Perspectives could also be defined for individuals who have special preferences or for specific kitchens. A perspective for the Smith's kitchen would include design rationale for its unique set of design decisions so that any future modifications could be checked for consistency with those decisions.

The organization of knowledge by perspectives encourages users to view the knowledge in terms of structured, meaningful categories which they can create and modify. It provides a structure of contexts that can correspond to categories meaningful in the design domain. This can ease the cognitive burden of manipulating large numbers of alternative versions of critics and other design knowledge.

Interpretive critics are the result of interactions between the perspectives structure and the critic mechanisms (Figure 7). Critics are associated with design perspectives. The perspectives provide a mechanism for creating, managing and selectively activating different sets of critics along with their related design knowledge, such as spatial relations, domain distinctions, palette items and argumentation. A perspective can incorporate critics from other perspectives, including generic and specific critics from the default perspective (see Figure 7). Additionally, a perspective may modify any inherited critics and define new ones.

Designers switch perspectives to examine a design from different viewpoints. Switching perspectives changes the currently effective definitions of critics, the terms used in these definitions, and other domain knowledge. As a result, the critics adapt to the different perspectives—hence the term "interpretive" critics. The designer always works within a particular perspective. At any time, the designer can select a different perspective by name. New perspectives can also be created by assigning a name and selecting existing perspectives to be inherited. Bob, the designer working with the Smiths in the previous scenario, could create a Smith's kitchen perspective and select the resale perspective to be inherited by it.

Perspectives are connected in an inheritance network; a perspective can modify any knowledge inherited from its parents or it can add new knowledge. Consider the inheritance network shown in Figure 7. Suppose that in the default perspective there is a rule that checks "if the top cabinets are 150 cm above the floor". In the Smith's kitchen perspective the rule that determines cabinet height is based on the cook's height. This *same* critic rule will be evaluated differently in the three different perspectives because it is defined in terms of the spatial relationship whose definition varies. Similarly, either the rule or the spatial relationship in the rule could be defined indirectly in terms of something in the argumentation issue-base, such as the answer to an issue requesting the primary cook's height. Critics and the design knowledge on which they are based can be adapted to interpret designs differently in many ways: by inheritance, by modification of inherited objects, or by addition of new objects into a perspective.

Interpretive critics based on perspectives provide a mechanism for refining the critiquing process that is orthogonal to the specific critics. Specific critics fine-tune the generic critics that embody general domain knowledge, relating them to the design choices specified for a given project. Whereas the set of generic and specific critics may be extensible in the sense that new critics can be added from time to time, the perspectives mechanism provides for multiple definitions of these sets to exist simultaneously so that individual designers can fluidly adopt varying viewpoints on designs. This provides a means for structuring new critics and other knowledge representations as they emerge during use of the design environment and systematically retaining this knowledge for use in future projects.

5 Benefits of embedding: increasing the shared context

Computational media offer great capacity for storing large volumes of information and support for managing dynamic information spaces (Norman, 1993). Computational media can integrate diverse information sources such as reference materials, solutions to previous design problems, and collections of design rationale. However, access to large information spaces creates a new problem for designers; information overload. In situations of information overload, the critical resource for designers is not information, but rather the attention with which to process information. Simon (1981) argued with convincing examples that a design representation suitable for a world in which the scarce factor is information may be exactly the wrong one for a world in which the scarce factor is attention. When presenting people with information, the primary concern is to present items that are relevant to the task at hand (Fischer & Nakakoji, 1991). Critics embedded in design environments exploit a rich notion of the designer's task at hand, or context, to provide relevant information to designers.

Design environments support a cooperative problem-solving process in which the designer determines the context of design by manipulating interface objects (such as graphical objects and form-based objects) in the construction, specification and perspective components. Objects in the construction component define a construction context that provides generic critics with a representation for the task at hand. Values and priorities for specification objects define a specific context that allows specific critics to compute relevant information for the particular task as specified by the designer. The perspective mechanism determines an interpretive context that enables collections of critics and their associated argumentation.

The context defined by the construction, specification and perspective situations allows the system to provide information relevant to a dynamic representation of the task at hand that is shared by the designer and the design environment. This shared context enables precise intervention by critics, reduces annoying interruptions, and increases the relevance of information delivered to designers. Critics embedded in design environments benefit the design process by increasing the designer's understanding of design situations, by pointing out significant design situations that might have been overlooked, and by locating relevant information in very large information spaces.

5.1 Increasing the designer's understanding of design situations

The solution of a design problem necessarily involves coming to a deeper understanding of the problem through attempts to solve it. Design problems cannot be clearly defined "up front", before any attempt at a solution is made. New requirements emerge during the design process (Schoen, 1983; Rittel, 1984; Fischer et al., 1992) that cannot be identified until portions of the artifact have been designed or implemented. These aspects of design create the following dilemma: (1) one cannot gather information meaningfully unless the problem is understood; (2) one cannot understand the problem without having a concept of the solution in mind; and (3) one cannot understand the problem without information about it.

Problem framing and problem solving are *mutually enabling* design processes because each informs the other. Design methodologists such as Schoen (1983) and Rittel (1984) stress the strong interrelationship between problem framing and problem solving. They characterize design problems by the need for designers to impose a discipline, or framing, on the problem to reduce the complexity of the situation to a manageable level. Problem framing is the process of determining the boundaries (or framework) of a problem, such as determining the "givens" of the problem, the assumptions under which the designer operates, and the criteria for evaluating a solution. Each move toward a design solution tests the problem framing, potentially exposing conflicting or unrealistic goals. Critics embedded in design environments support designers in creating and modifying the problem framing throughout the design process—not just in the beginning. Critics support a design process where "understanding the problem is the problem".

In this view of design, in which problem framings and problems solutions coevolve, each action by the designer has the potential to alter the understanding of the problem, which in turn can influence subsequent actions. Our goal is to support design as a cooperative problem-solving dialogue between the designer and the evolving dèsign situation.

5.2 Pointing out significant design situations

By seeing design as a "reflective conversation with the situation" (Schoen, 1983), action is governed by nonreflective thought processes and proceeds until it breaks down. A *breakdown* (Fischer, 1993) occurs when the designer realizes that nonreflective action has resulted in unanticipated consequences—either good or bad. Schoen described this realization as "the situation talks back". Reflection is used to repair the breakdown, and then (nonreflective) situated action continues. The hallmark of reflection-in-action is that it takes place within the *action present*—within the time period during which the decision to act has been made but the final decision about how to act has not. This is the time period during which reflection can still make a difference in what action is taken.

Schoen's theory of design is based on designers interacting with traditional media, and the back-talk from the situation is determined solely by the designer's skill, experience and attention. Computational technology, such as critics embedded in design environments, afford a new type of back-talk from the design situation. Computational design situations can actively point out breakdowns to designers. This active design support enables designers to hear the situation talk back in situations that might have remained mute in passive media.

Reflection-in-action, as supported by embedded critics, is an ongoing cycle of action, breakdown and reflection. Designers act when they shape the design situation. They establish a shared context with the design environment by manipulating interface objects in the construction, specification or perspective components. Breakdowns are triggered by critics embedded in the design environment that detect situations that indicate the designer might need to reflect. Based on the shared context, critics support reflection by delivering information relevant to the breakdown situation. Argumentative information helps designers understand the breakdown situation, and the catalog contains design solutions that provide examples of how other designers have resolved similar problems.

The scenario illustrates how embedded critics support design as a reflective conversation with the situation. In the scenario, critics triggered two consecutive breakdowns. In the first, the construction situation talked back to Bob when his actions violated a generic kitchen design principle that "the dishwasher should not be too far from the sink". After some reflection, he moved the dishwasher nearer to the sink to comply with the critic. However, this action created a new breakdown situation. A specific critic signalled a breakdown to remind Bob that his actions were inconsistent with his partial specification; that is, his placement of the sink might not be optimal for left-handed cooks. This breakdown led him to reflect on his goals; instead of altering the design construction, Bob reformulated his partial specification.

5.3 Locating relevant information in large information spaces

Making information relevant to the task at hand poses many challenges for the design of interactive computer systems, particularly for problems in which the need for information is critical and yet precise information needs cannot be known in advance of attempts to solve the problem. Our design environments that support design in complex domains are high-functionality computer systems; that is, they provide a large amount of functionality and are built on large information bases. Such systems provide more information and functionality than a single person can master (Draper, 1984). Two factors contribute to this behaviour: (1) the effort of finding information often outweighs the perceived benefits of doing so; and (2) users are not aware that the information even exists. Both factors can be related to the discrepancy between the designer's perception of an information space and the actual information contained in a high-functionality system (see Figure 8).

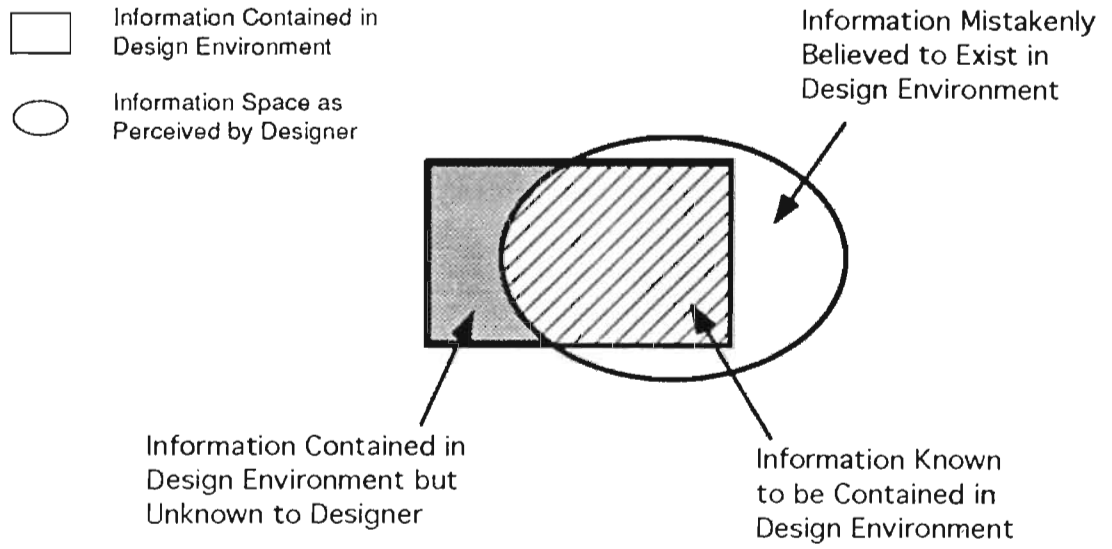

Figure 8 Large information spaces contain more information than a single person can know exists. The oval represents the information a designer perceives to be in the design environment. The square represents the information actually contained in the design environment. This figure illustrates that the designer's perception includes information that does not exist in the design environment, and does not include some information that actually exists in the design environment.

Designers are often unwilling to disrupt the design process to search for information in large information spaces, even if they know the information exists. In addition, designers may not know when they *need* information. Embedded critics save designers the trouble of explicitly querying the system for information. Critics notify designers of situations indicating the need to reflect (breakdowns) and provide access to information fueling reflection. The context of the breakdown situation serves as an implicit query that enables embedded critics to deliver relevant information. Designers benefit from needed information without having to explicitly ask for it.

Embedded critics can also deliver relevant information (Nakakoji, 1993) about which designers were unaware (see Figure 8). Critics provide the designer with a pointer into part of the system's information space with which the designer needs to become aware. The designer can further browse the unfamiliar portion of the information space starting from the entry point provided by the critic.

Critics afford *learning on demand* (Fischer, 1991) by letting designers access new knowledge in the context of actual problem situations; users are informed (1) when they are getting into trouble, (2) when they are missing important information, and (3) when they come up with problematic solutions. Learning on demand is a promising approach for the following reasons: (1) it contextualizes learning by integrating it into work rather than relegating it to a separate design phase; (2) it lets designers see for themselves the usefulness of new knowledge for actual problem situations, thereby increasing the designers' understanding of their situations; and (3) it makes new information relevant to the task at hand, thereby leading to better decision making, better products, and better performance.

Critics exploit the shared context of breakdown situations to compute what information is relevant to the task at hand. In the scenario, each critic's notification message was linked to information in the argumentation component. For the "dishwasher not too far from the sink" issue, the designer was reminded of plumbing requirements he might have known about but did not remember in the context of the design situation. The "left-handed" specific critic identified information the designer had previously been unaware of: that the recommended positions of the sink and dishwasher are dependent upon whether the cook is right- or left-handed. The interpretive critic (enabled by adapting a Resale Perspective) informed Bob of additional information about which he had previously been unaware. Now that he is aware of this "resale" value concern, Bob

could explore further implications of a resale perspective by browsing related information or by continuing his design process, where he will be informed on demand.

6 The dynamic nature of critiquing knowledge

6.1 Supporting designers in adapting the critiquing system

To be successful, embedded critiquing systems must adapt to reflect changes in the design domain. Two questions arise when considering system adaptation: will designs be *able* to adapt the system as required, and will designers be *motivated* to adapt the system? End-user modifiability components and design environment "seeds" are important steps toward answering these questions.

Adapting the critiquing system involves modifying or adding critic rules, design units, design unit relations and critic explanations in the form of argumentation and catalog examples. Sometimes, adapting the system is as simple as changing parameters or filling out specialized forms. Girgensohn (1992) explored end-user modifiability in domain-oriented design environments. His work showed that end-users without any formal training in computer science need considerable environmental support in the form of explanatory help, critics that support modification processes, task decomposition agendas, and computer-supported object classification to effect significant system changes. Even with this extensive environmental support, none of the subjects in his user studies were able to complete the adaptations without intervention from the study supervisor. Girgensohn's research has demonstrated that enabling designers to adapt their systems is a very difficult problem which requires further research in the areas of demonstration components, domain-oriented knowledge representations, and adaptive user modelling components. The HERMES project is exploring a different approach toward achieving end-user modifiability by building into the design environment an English-like end-user programming language (Stahl et al., 1992).

6.2 "Seeding" the critiquing system with domain knowledge

Whereas ongoing adaptation of embedded critiquing systems is in the hands of designers solving design problems, system builders must create the original conditions that enable and motivate this evolution process to occur. Specifically, system builders must provide initial environments in the form of a seed.

We cannot offer an easy-to-follow prescription for successful seed building. Seed building requires a deep understanding not only of the application domain, but also of the *practice* (Ehn, 1989) of the people who will use the system. System builders cannot hope to attain such an understanding without, at least to some extent, becoming domain experts themselves. But this is generally infeasible. For useful seeds to be built, system-building must be based on a process of mutual education (Greenbaum & Kyng, 1991) between system builders, who know about building software design environments, and domain designers, who understand the practice of design in the target application domain. The goal of this mutual education process is to establish a shared understanding of what domain knowledge a seed should contain so that it will immediately support the practice of designers within that domain.

6.3 Accumulating design knowledge through critics

Embedded critics play the crucial role of "knowledge attractors" in domain-oriented design environments. Design knowledge surfaces during reflection-in-action, when designers reflect upon the source of breakdowns and devise courses of action for resolving the breakdowns. User observations in using specific critics revealed that when designers were fired a critic rule, they often argued for or against the associated argument and were motivated to describe the reason by articulating pro or counter arguments to the argumentation (Nakakoji, 1993). The incomplete nature of design knowledge guarantees the argumentation is never complete. Designers who arrive

at an innovative resolution to a breakdown may add their arguments to the existing rationale, enriching the information space contained in the design environment.

7 Conclusions

Although this paper focuses primarily on a single design environment built for residential kitchen design, the HYDRA-KITCHEN system, other ongoing research in our group has demonstrated that embedded critiquing systems have broad applicability to a variety of domains and that embedded critiquing systems can be applied to complex, new domains with few accepted design rules and practices, and non-spatially-oriented domains.

The interpretive critiquing mechanism is being explored in the domain of lunar habitat design (Stahl, 1993). Unlike kitchen design, lunar habitat design is a completely new domain with few design rules and no standardized vocabulary. In domains with few standards, negotiation, argumentation and interpretation are increasingly important aspects of design. This aspect of the lunar habitat design domain led us to extend our critiquing systems to include interpretive mechanisms.

The Voice Dialog Design Environment tests the applicability of critiquing systems to non-spatial domains. The system supports the design and simulation of applications with phone-based interfaces (Repenning & Sumner, 1992). In this domain, design units include audio prompts, voice menus and telephone touch-tone input. Relations between design units are temporal in nature; that is, design units occur before or after certain events in the execution sequence. This design environment is part of a joint research project between the University of Colorado and voice dialogue application designers at US WEST Advanced Technologies (Sumner et al., 1991).

We have demonstrated how embedding critic mechanisms in design environments overcomes many deficiencies found in stand-alone critiquing systems. The generic, specific and interpretive critics we have explored correspond to three dimensions of embedding. Generic critics are embedded in the construction context because they are enabled by the placement of design units in the work area. Specific critics are embedded in the partial specification by being dynamically constructed from domain distinctions tied to specification items; they reduce the intrusiveness of generic critics by narrowing the enabled critics to those that are relevant to the partially specified task at hand. Interpretive critics are embedded in the network of perspectives that supports the evolution of alternative viewpoints on designs; using these critics, designers are able to consider their designs critically from multiple perspectives. The beneficial role of human critiquing in science, design and engineering had been socially recognized long before the advent of computational critiquing systems. Our approach of embedding critics into integrated design environments is an important step toward applying the critiquing paradigm to create more useful and usable knowledge-based computer systems.

Acknowledgements

The authors thank the members of the Human-Computer Communications group at the University of Colorado, who contributed to the conceptual framework and the system discussed in this paper. The research was supported by the National Science Foundation under grants No. IRI-8722792 and IRI-9015441, by the Colorado Advanced Software Institute, by US WEST Advanced Technologies, by the NYNEX Science and Technology Center, and by Software Research Associates, Inc. (Tokyo, Japan). We especially wish to thank Barbara Gibson at Kitchen Connection in Boulder, Colorado, for sharing her expertise in kitchen design.

References

Bobrow, DG and Goldstein, I, 1980. "Representing design alternatives" In: *Proceedings of AISB Conference*, AISB, Amsterdam.

Buchanan, B and Shortliffe, E, 1984. "Human engineering of medical expert systems" In: B Buchanan and E Shortliffe (Eds), *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, Addison-Wesley, Reading, MA, pp 599–612.

Burton, R and Brown, JS, 1982. "An investigation of computer coaching for informal learning activities" In: D Sleeman and JS Brown (Eds), *Intelligent Tutoring Systems*, Academic Press, London, pp 79–98.

Chambers, AB and Nagel, DC, 1985. "Pilots of the future: Human or computer?" *Commun. ACM* **28** (11) 1187–1199.

Conklin, J and Begeman, M, 1988. "gIBIS: A hypertext tool for exploratory policy discussion" *Trans. Office Infor. Syst.* **6** (4) 303–331.

Draper, SW, 1984. "The nature of expertise in UNIX" In: *Proceedings of INTERACT'84, IFIP Conference on Human-Computer Interaction*, pp 182–186, Elsevier, Amsterdam.

Ehn, P, 1989. *Work-Oriented Design of Computer Artifacts* (2nd ed.), Arbetslivscentru, Stockholm.

Fischer, G, 1987. "A critic for LISP" In: *Proceedings of the 10th International Joint Conference of Artificial Intelligence*, pp 177–184, Milan, Italy.

Fischer, G, 1990. "Communication requirements for cooperative problem solving systems" *Information Syst.* **15** (1) pp 21–36.

Fischer, G, 1991. "Supporting learning on demand with design environments" In: *Proceedings of the International Conference on the Learning Sciences*, pp 165–172, Evanston, IL.

Fischer, G, 1992. "Domain-oriented design environments" In: *Proceedings of 7th Annual Knowledge-Based Software Engineering (KBSE-92) Conference*, pp 204–213, McLean, VA.

Fischer, G, 1993. "Turning breakdowns into opportunities for creativity" In: E Edmonds (Eds), *Creativity in Cognition*, Penrose Press.

Fischer, G and Girgensohn, A, 1990. "End-user modifiability in design environments" In: *Proceedings of CHI'90*, pp 183–191, ACM Press.

Fischer, G, Grudin, J, Lemke, AC, McCall, R, Ostwald, J, Reeves, BN and Shipman, F, 1992. "Supporting indirect, collaborative design with integrated knowledge-based design environments" *Human Computer Interaction* (Special Issue on Computer Supported Cooperative Work) **7** (3).

Fischer, G, Lemke, AC, Mastaglio, T and Morch, A, 1991. "The role of critiquing in cooperative problem solving" *ACM Trans. Infor. Syst.* **9** (2) 123–151.

Fischer, G, Lemke, AC, McCall, R and Morch, A, 1991 "Making argumentation serve design" *Human Computer Interaction* **6** (3–4) 393–419.

Fischer, G, Lemke, AC and Schwab, T, 1985. "Knowledge-Based Help Systems" In: *Proceedings of Human Factors in Computing Systems, CHI'85 Conference Proceedings*, pp 161–167, San Francisco, CA.

Fischer, G, McCall, R and Morch, A, 1989. "Design environments for constructive and argumentative design" In: *Proceedings of CHI'89*, pp 269–275, ACM Press.

Fischer, G and Nakakoji, K, 1991. "Making design objects relevant to the task at hand" In: *Proceedings of AAAI-91, Ninth National Conference on Artificial Intelligence*, pp 67–73, AAAI Press/The MIT Press.

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

Greenbaum, J and Kyng, M, 1991. *Design at Work: Cooperative Design of Computer Systems*, Lawrence Erlbaum.

Hackman, JR and Kaplan, RE, 1974. "Interventions into group process: An approach to improving the effectiveness of groups" **5** 459–480.

Johansen, R, 1988. *Groupware: Computer Support for Business Teams*, Free Press.

Kolodner, J, 1991. "Improving human decision making through case-based decision aiding" **12** (2) 52–68.

Lave, J, 1988. *Cognition in Practice*, Cambridge University Press.

Lemke, AC and Fischer, G, 1990. "A cooperative problem solving system for user interface design" In: *Proceedings of AAAI-90, Eighth National Conference on Artificial Intelligence*, pp 479–484, AAAI Press/ The MIT Press.

McCall, R, 1987. "PHIBIS: Precedurally Hierarchical Issue-Based Information Systems" In: *Proceedings of the Conference on Architecture at the International Congress on Planning and Design Theory (New York)*, American Society of Mechanical Engineers.

Nakakoji, K, 1993. *Increasing shared knowledge of design tasks between humans and design environments: The role of a specification component*, PhD Dissertation Thesis, Department of Computer Science, University of Colorado at Boulder.

Norman, DA, 1993. *Things That Make Us Smart*, Addison-Wesley.

Petroski, H, 1985. *To Engineer is Human: The Role of Failure in Successful Design*, St. Martin's Press.

Polanyi, M, 1966. *The Tacit Dimension*, Doubleday.

Popper, KR, 1965. *Conjectures and Refutations*, Harper & Row.

Prieto-Diaz, R and Freeman, P, 1987. "Classifying software for reusability" **4** (1) 6–16.

Repenning, A and Sumner, T, 1992. "Using agentsheets to create a voice dialog design environment" In: *Proceedings of Symposium on Applied Computing (SAC'92)*, pp 1199–1207, ACM Press.

Rittel, H, 1984. "Second generation design methods" In: N Cross (Ed.) *Developments in Design Methodology*, pp 317–327, Wiley.

Rittel, H and Webber, MM, 1984. "Planning problems are wicked problems" In: N Cross (Ed.) *Developments in Design Methodology*, pp 134–144, Wiley.

Schoen, DA, 1983. *The Reflective Practitioner: How Professionals Think in Action,* Basic Books.

Silverman, B, 1992. "Survey of expert critiquing systems: Practical and theoretical frontiers" *Comm. ACM* **35** (4) 106–127.

Simon, HA, 1981. *The Sciences of the Artificial* (2nd ed.), The MIT Press.

Stahl, G, 1992. *Toward a Theory of Hermeneutic Software Design*, No. CU-CS-589-92, Computer Science Department, University of Colorado at Boulder.

Stahl, G, 1993. "Supporting interpretation in design" *J. Architecture and Planning Research* (Special Issue on Computational Representations of Knowledge) (Forthcoming).

Stahl, G, McCall, R and Peper, G, 1992. "A hypermedia inference language as an alternative to rule-based expert systems" (Submitted to Expert Systems ITL Conference).

Sumner, T, Davies, S, Lemke, AC and Polson, PG, 1991. *Iterative Design of a Voice Dialog Design Environment*, Technical Report No. CU-CS-546-91, Department of Computer Science, University of Colorado at Boulder.

Winograd, T and Flores, F, 1986. *Understanding Computers and Cognition: A New Foundation for Design*, Addison-Wesley.