EMBEDDING COMPUTER-BASED CRITICS IN THE CONTEXTS OF DESIGN

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

[1]University of Colorado at Boulder
Department of Computer Science
Boulder CO USA 80309-0430
(303) 492-1592
E-mail: gerhard@cs.colorado.edu

[2]Software Research Associates
Software Engineering Laboratory
1-1-1 Hirakawa-cho, Chiyoda-ku
Tokyo 102, Japan
E-mail: kumiyo@cs.colorado.edu

ABSTRACT
Computational critic mechanisms provide an effective form of computer-human interaction supporting the process of design. Computational critics embedded in domain-oriented design environments can take advantage of knowledge representations in these environments to provide less intrusive, more relevant critiques. Three classes of embedded critics have been studied. *Generic critics* use domain knowledge to detect problematic situations in the design construction. *Specific critics* take advantage of additional knowledge in the partial specification to detect inconsistencies between the design construction and the design specification. *Interpretive critics* are tied to a perspective mechanism that supports designer's in examining their artifact from different viewpoints.

KEYWORDS: Generic critics, specific critics, interpretive critics, design environments, specification, construction, domain orientation, perspectives, critiquing systems.

INTRODUCTION
We view design as a process of successive refinement through trial, breakdown, interpretation, and reflection. Critiquing – the communication of a reasoned opinion about an artifact or a design – plays a central role in the design process. Computational critic mechanisms provide an effective form of computer-human interaction supporting this important aspect of design. We have developed a series of prototype computer-based design environments containing computational critiquing mechanisms to investigate the how such environments can provide timely and relevant knowledge to designers.

Our research group's early work focused on building and evaluating stand-alone critiquing mechanisms. Critical analyses of these and other systems [6], [14], combined with empirical evaluations, led us to realize that the challenge in building critiquing systems is not simply to provide design feedback: the challenge is to say the "right" thing at the "right" time. We claim that embedding critics in domain-oriented design environments has provided an effective response to this challenge. Design environments are computer programs which support designers in concurrently specifying a problem, constructing a solution, and interpreting an emerging design from alternative perspectives. Embedded critics can provide more focused, less intrusive critiques by taking advantage of knowledge of the contexts of design: the domain, the construction situation, the partial specification, and interpretive perspectives.

While we have investigated critiquing in numerous domains such as computer network design and lunar habitat design, the examples for this article will be based on floor plan design for kitchens. This paper first describes the evaluations and theoretical motivations that led to the redesign of our critiquing mechanisms. We analyze early systems and empirical results that exposed the deficiencies of stand-alone critiquing mechanisms. Next, we present our redesign: three classes of embedded critics – generic, specific, and interpretive critics. Finally, we conclude with a discussion of the benefits of this new approach.

ANALYSIS OF EARLY CRITIQUING SYSTEMS
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 facility 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.

Saying the "right" thing...
LISP-CRITIC [2] 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). The lack of domain orientation of LISP-CRITIC limits the depth of critical analysis the critiquing system can provide. Without domain knowledge, critic rules cannot be tied to higher level concepts; i.e., LISP-CRITIC can only answer questions such as whether the Lisp code can be written more efficiently, but it cannot assist the user in deciding whether the code can solve the user's problem.

FRAMER [11] 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 that many users did not understand the consequences of following the critic's advice or why the advice was beneficial to solving their problem. We have observed that when users do not understand why a suggestion is made, they tend to follow the critic's advice whether or not it is appropriate to their situation. FRAMER II [10] provided short explanations to address this problem. However, in design there are not always simple answers; access to argumentative discussions are necessary [12].

JANUS [5] is a step towards 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 issues. These critics often give spurious or irrelevant critic advice. These irrelevant suggestions result 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. With an explicit model of the designer's intentions for a particular design, critics can be selectively enabled and provide less intrusive and more relevant advice.

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 [9] was developed to respond to this problem by making the domain knowledge (including critics) end-user modifiable. However, 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 looking at a problem from many perspectives (technical, structural, functional, aesthetic, personal), and critiquing accordingly.

... at the "right" time

Research with these and other systems [7], [1] also investigated critic intervention strategies, i.e., strategies determining 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 [11] 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 prohibitively 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 experience with our early critiquing systems 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. What is needed is a strategy 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 implicitly control critic intervention through the designer's interaction with the construction, specification, and interpretation design contexts.

THEORETICAL MOTIVATION

Our evaluations of computer-based critiquing mechanisms show that while critics provide useful support for people engaged in design tasks, a number of problems arise if the critics are not adequately attuned to the task at hand. Design methodologists and proponents of situated cognition have argued that human critical reflection during designing is *situated* in various ways, suggesting that computational critics should be made similarly context-dependent.

Suchman [17] argues that when pursuing a task, people do not necessarily follow an explicit step-by-step plan they have mentally worked out ahead of time. Rather, they respond to their changing environment based on tacit skills. Schoen [13] describes design as a process of reflection-in-action where each design move creates a new situation, which may challenge the assumptions and strategies under which the designer is operating. These situations signal the designer of a need to reflect upon the design situation and possibly to formulate new strategies.

Another approach is suggested by Rittel [12], who sees design as a process of argumentation. A domain like kitchen design consists of a variety of issues to be resolved in completing a task. Within the context of a specific design project, arguments for various answers to these issues can be debated from many perspectives. So solutions are not dependent only upon the unique task, but also upon the background, interests, and commitments of the various stakeholders: i.e., the designers, their clients, and the eventual users. More generally, Winograd and Flores [18] stress the role of interpretation in design. Designers interpret the task, the consequences of possible design decisions, and competing design rationale from their shared or individual perspectives.

These theorists reject the rationalist waterfall model according to which designers first derive an exhaustive specification of a task and then proceed to methodically implement the specification. Rather, design is an integrated process of problem solving (design construction) and problem framing (task specification).

These theoretical considerations suggest that critical reflection is most effective when embedded in a number of inter-dependent contexts. To support critical reflection in design, we embed critiquing mechanisms in design environments representing a variety of contexts. First, there is the context of knowledge of the *domain* itself. We represent this as an issue-base capturing the accepted wisdom of the field, a catalog of illustrative past designs, and a palette of domain-oriented components. Unlike the rule-base of an expert system, the issue-base is neither complete nor consistent, but supports design as an argumentative process by incorporating alternative and opposed viewpoints. Second, we represent the current state of *construction*. For many domains, a graphical display of architectural or structural layout is convenient and useful. Third, the evolving partial *specification* is included to guide evaluation of the adequacy of design. For our kitchen design system, we mimic the format of professional kitchen designers' questionnaires. Finally, support is provided for the definition of group and personal versions of domain knowledge that can represent critical *interpretations*. By embedding critics in the contexts of the domain, construction, specification and interpretation, we overcome the problems of stand-alone critics.

EMBEDDING CRITICS IN DESIGN ENVIRONMENTS

We have responded to our evaluation of early critiquing mechanisms and to the theoretical arguments for contextualization by embedding critics in computational design environments. We have explored three types of critiquing mechanisms: *generic*, *specific*, and *interpretive* critics. These mechanisms are described below in a scenario involving HYDRA [4], a new version of the kitchen design environment (Figure 1).

An Integrated Design Environment

Reflection on the shortcomings of JANUS [5] led us to extend it by incorporating representations of several aspects of the design context. Like JANUS, HYDRA contains both a construction and an argumentation component. HYDRA also supports a specification component and a catalog of designs. The specification is based on questionnaires used by professional kitchen designers to elicit their customer's requirements, such as the kitchen owner's cooking habits and family size [8]. The catalog is a repository for past designs that are illustrative of the possible design space. Catalog entries support case-based reasoning and provide concrete design examples of issues discussed in the argumentation component. These software components of the HYDRA system provide design creation tools and information repositories which reflect the real-world contexts of the design process.

Embedding critiquing systems in integrated design environments has several benefits. First, they have an increased level of critical analysis because critiquing mechanisms have been tied to the partial construction and the domain knowledge. The argumentation base and catalog of designs provide rich sources of domain knowledge that the critiquing mechanism can use in its explanation process. Second, the specification component provides an explicit representation of the designer's intentions for the design. The critiquing mechanism can take advantage of this information to enable sets of critics to evaluate the current design construction selectively for adherence to the designer's stated goals. Third, critiquing can be done from specific viewpoints, such as construction costs, resale value, plumbing

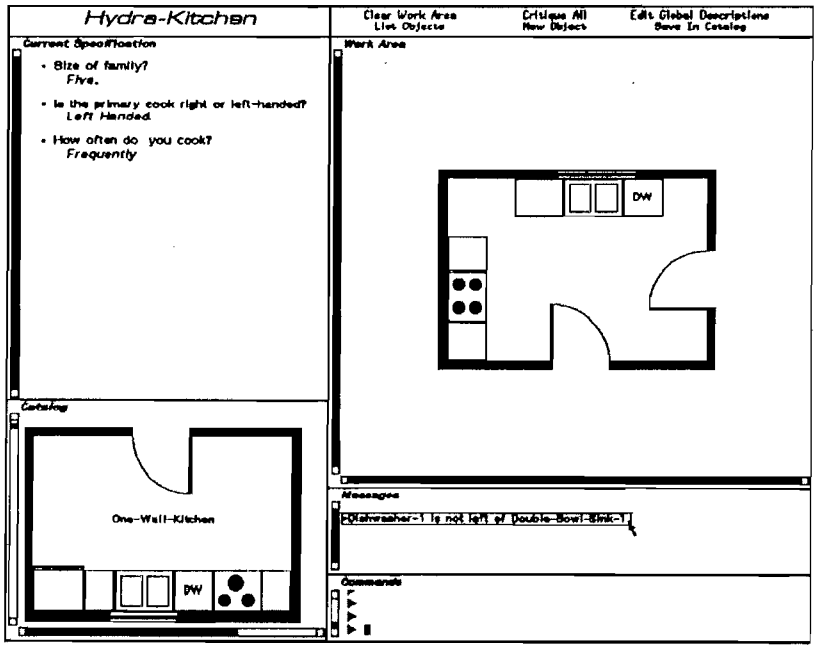

Figure 1. HYDRA-KITCHEN consists of a work area for design construction, a specification area, a message window for critic notifications, and a catalog of illustrative design examples.

concerns, or work flow. Personal and group perspectives can also be developed to provide critiquing from different cultural, socio-economic, or idiosyncratic viewpoints.

Scenario Illustrating Generic, Specific and Interpretive Critics

Bob has been asked to design a kitchen for the Smith family. Working with the Smiths, Bob enters the partial specification shown in Figure 1.

Bob begins working on a floor plan in the HYDRA construction work 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 decides 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.

This action triggers a *specific critic* with the message, "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 enabled because Bob specified that the primary cook is left-handed. 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 remodeling 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 message "The dishwasher should be on the right side of the sink." Interpretive critics support design as a hermeneutic 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 stove 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 most 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. This will also provide access to argumentation concerning designing for resale practices.

Three Embedded Critiquing Mechanisms

Embedded critics integrate the components of the design environment. 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 analyzed for compliance with the currently enabled set of critic rules; (3) when a lack of compliance is detected, the critic signals a breakdown

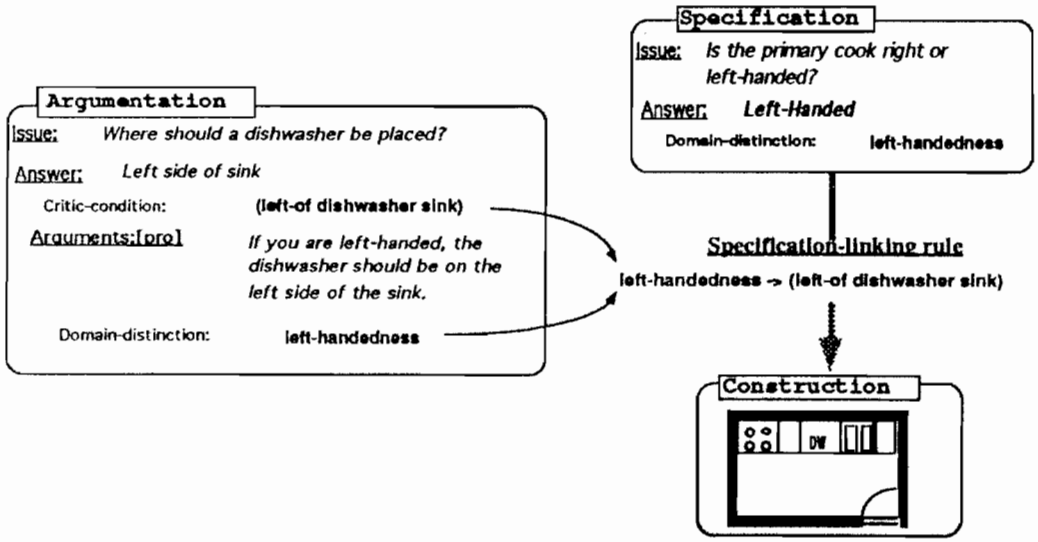

Figure 2. Derivation of the Specification-Linking Rules and illustration of critic parts – "left-hand dishwasher " . The domain distinction associated with a specification item is paired with a matching pro or con argument in the argumentation base. The Lisp expression for the associated answer is linked with the domain distinction to form a specific critic rule. The domain distinction associated with the specification item shown in the figure is "left-handed-cook."

and provides entry into the exact place in the argumentative hypermedia system where the appropriate explanation is located; (4) concrete catalog examples that illustrate the explanation given in the form of argumentation can optionally be delivered [3].

Generic critics. All three critic mechanisms – generic, specific, and interpretive – use a production system style of knowledge representation and follow the basic critiquing process described above. Critic rules consist of condition and action clauses plus links into the argumentation context. The *condition* clause checks whether a certain situation exists in the current design construction and is defined in terms of spatial relations between design units, such as near, far, next-to, etc. The *action* clause notifies the designer that a particular situation has been detected.

Each critic rule is linked to a particular issue in the argumentation base. The designer can view the critic's associated argumentation by selecting the initial notification message to display an entry-point into the hypermedia issue-base. Such argumentative explanations help designers determine why the design situation identified by the critic message may be significant or problematic. Designers can optionally explore the issue-base or select an issue and an associated answer in the argumentation and request to see a positive example or a counter-example from the catalog of designs.

The three types of embedded critics differ from one another in how they determine which set of critic rules should be enabled. Generic critics evaluate the construction situation based on an assumption that a designer wants to design a "good" kitchen as the default design requirement.

Specific Critics. Specific critics evaluate the construction situation for compliance with the partial specification. Specific critics are constructed dynamically from the partial specification to reflect current design goals. Specification-linking rules [8] link domain distinctions activated in the specification with appropriate critic conditions. These critic conditions are specialized by adding requirements to reflect the current specification. Domain distinctions are a vocabulary for expressing domain concepts, like left-handedness, safety, and efficiency. Each specification item is associated with one or more domain distinctions. Whenever the designer modifies the specification, the critiquing system recompiles the specification-linking rules. (See Figure 2.) In this way, critiquing criteria are tied to a representation of the partially articulated goals of a specific design project.

Interpretive Critics. Interpretive critics [15], [16] provide support for design as a hermeneutic (interpretive) process. They allow designers to interpret the design situation according to their interests. Interpretive critics are associated with design perspectives rather than with partial specifications. Perspectives provide a mechanism for creating, managing, and selectively activating different sets of critics and design knowledge, such as spatial relations, domain distinctions, palette items, and argumentation. Perspectives are arranged in an inheritance hierarchy; a child perspective can modify knowledge inherited from its parent perspective or it can add new knowledge.

Designers switch perspectives to examine a design from different viewpoints. This changes the currently effective definitions of critics, the terms used in these definitions, and other domain knowledge (Figure 3).

DISCUSSION

Embedded critics represent another iteration cycle in our continuing research into computer-based design environments and critiquing systems. Embedded critics were designed and built in response to deficiencies uncovered in our early critiquing systems (LISP-CRITIC, FRAMER, JANUS), as well as insights gained from design theorists (Schoen, Rittel, Ehn) and situated cognition researchers (Suchman, Winograd, Lave).

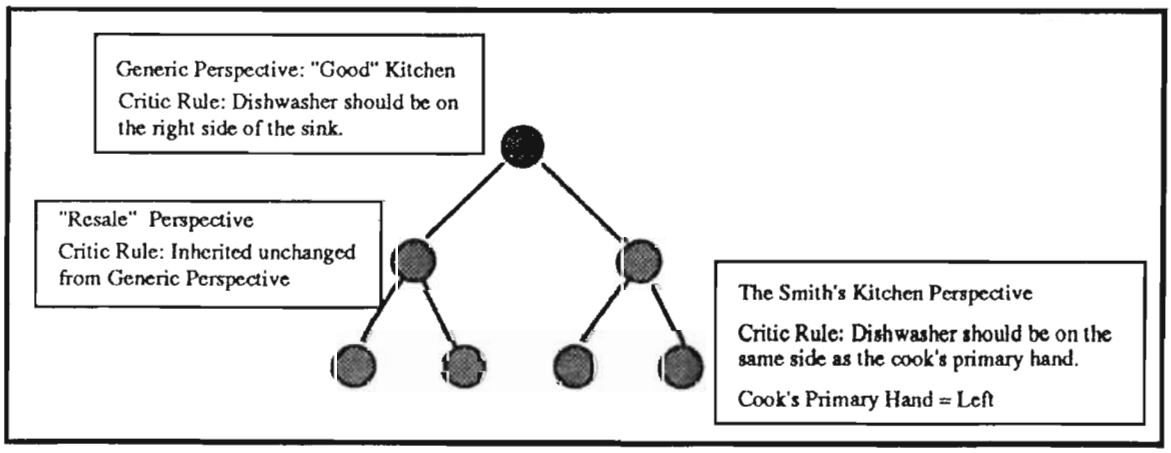

Figure 3. Design contexts are arranged in an inheritance hierarchy. Three perspectives - the generic, the resale, and the Smith's – are shown. The preferred placement of the dishwasher depends on the perspective selected.

Design environments support designers in creating and modifying the problem setting throughout the design process, not just in the beginning. Problem setting in design environments is supported by the specification component, where designers articulate their goals and priorities for the design. The problem setting as represented in the partial specification does not serve as a rigid template for constructing a solution, but rather as a flexible framework in which to operate. Embedded critics support the integration of problem setting and problem solving by making explicit relationships between the partial specification and the construction situation. Embedded critics evaluate the construction situation for compliance with the partial specifications, within a chosen perspective. When critics detect a conflict, the need to reflect-in-action is signaled to the designer. Resolving the breakdown might require that the specification be modified (the problem re-framed) or that the construction situation be modified to eliminate conflicts.

The three classes of critics we have explored correspond to three dimensions of embedding. Generic critics are embedded in the construction, because they are enabled by the placement of design units in the work area. Specific critics are embedded in the partial specification, in that they are dynamically constructed from domain distinctions tied to specification items. Specific critics 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 hierarchy of perspectives that allow the evolution of alternative viewpoints on designs. Using these critics, designers are able to consider their designs critically from multiple viewpoints.

Our approach of embedding critics in integrated design environments is an important step towards applying the critiquing paradigm to create more useful and usable knowledge-based computer systems. We have argued that embedded critics focus the attention of the system on the concerns of the designer in order to deliver the "right" thing at the "right" time. Future research will focus on evaluating embedded critiquing systems in naturalistic settings, i.e., observing the systems in use by professional designers in their regular design activities.

ACKNOWLEDGMENTS

The research was supported by the National Science Foundation under grants No. IRI-8722792 and MDR-9253425, by the Colorado Advanced Software Institute under grants in 1990/91, 1991/92, 1992/93, by US West Advanced Technologies, by NYNEX Science and Technology Center, and by Software Research Associates, Inc. (Tokyo, Japan).

REFERENCES

[1] R. Burton and J. S. Brown, "An Investigation of Computer Coaching for Informal Learning Activites," in *Intelligent Tutoring Systems*, D. Sleeman and J. S. Brown, Ed., London, Academic Press, 1982, pp. 79-98.

[2] G. Fischer, "A Critic for LISP," *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, Milan, Italy, 1987, pp. 177-184.

[3] G. Fischer, "Communication Requirements for Cooperative Problem Solving Systems," *Informations Systems*, Vol. 15, pp. 21-36, 1990.

[4] G. Fischer, A. Girgensohn, K. Nakakoji and D. Redmiles, "Supporting Software Designers with Integrated, Domain-Oriented Design Environments," *IEEE Transactions on Software Engineering*, Vol. 18, pp. 511-522, 1992.

[5] G. Fischer, A. Lemke, T. Mastaglio and A. Morch, "Using Critics to Empower Users," *CHI*, Seattle, WA, 1990, pp. 337-347.

[6] G. Fischer, A. C. Lemke, T. Mastaglio and A. Morch, "The Role of Critiquing in Cooperative Problem Solving," *ACM Transactions on Information Systems*, Vol. 9, pp. 123-151, 1991.

[7] G. Fischer, A. C. Lemke and T. Schwab, "Knowledge-Based Help Systems," *Human Factors in Computing Systems, CHI'85 Conference Proceedings (San Francisco, CA)*, , pp. 161-167, 1985.

[8] G. Fischer and K. Nakakoji, "Making Design Objects Relevant to the Task at Hand," *Proceedings of AAAI-91, Ninth National Conference on Artificial Intelligence*, , pp. 67-73, 1991.

[9] A. Girgensohn, "End-User Modifiability in Knowledge-Based Design Environments," *Ph.D.*, CU-CS-595-92, Computer Science Dept., University of Colorado at Boulder, 1992.

[10] A. C. Lemke, "Design Environments for High-Functionality Computer Systems," PhD Thesis, 1989.

[11] A. C. Lemke, "Cooperative Problem Solving Systems Must Have Critics," *Proceedings of the AAAI Spring Symposium on Knowledge-Based Human Computer Communication*, , pp. 73-75, 1990.

[12] H. Rittel, "On the Planning Crisis: Systems Analysis of the First and Second Generations," *Bedriftsokonomen*, Vol. 8, pp. 390-396, 1972.

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

[14] B. Silverman, "Survey of Expert Critiquing Systems: Practical and Theoretical Frontiers," *CACM*, Vol. 35, pp. 106-127, 1992.

[15] G. Stahl, "Toward a Theory of Hermeneutic Software Design," Computer Science Dept., University of Colorado at Boulder, *Tech Report* CU-CS-589-92, March 92.

[16] G. Stahl, "A Computational Medium for Supporting Interpretation in Design," *Journal of Architecture and Planning Research*, June 1993.

[17] L. Suchman, *Plans and Situated Actions: The Problem of Human-Machine Communication*, Cambridge University press, Cambridge, 1987.

[18] T. Winograd and F. Flores, *Understanding Computers and Cognition: A New Foundation for Design*, Addison-Wesley, Menlo Park, CA, 1986.