EMBEDDING COMPUTER-BASED CRITICS IN THE CONTEXTS OF DESIGN

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

ABSTRACT

Computational critiquing mechanisms provide an effective form of computer-human interaction supporting the process of design. Critics embedded in domain-oriented design environments can take advantage of additional knowledge residing in these environments to provide less intrusive, more relevant critiques. Three classes of embedded critics have been designed, implemented, and 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 perspective mechanisms that support designers in examining their artifact from different viewpoints.

K E Y W 0 R D S: 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 [15, 16, 18, 21]. 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 design environments containing critiquing mechanisms to investigate 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 [7, 17], 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 that 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 [5] and lunar habitat design [19], the exarnples for this article will be based on floor plan design for kitchens [6]. This paper first describes the evaluations and theoretical motivations that led to the redesign and extensions 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. We conclude with a discussion of the benefits of this new approach.

ANALYSIS OF EARLY CRITIOUING 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 [3, 8, 12] 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, 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 users in deciding whether the code can solve their problem.

FRAMER [13] 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 1 2] provided short explanations to address this problem. However, in design there are not always simple answers; access to argumentative discussions are necessary *[15].*

JANUS [6, 7] 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. 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. 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 [10] 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 interpreting a problem from many perspectives (technical, structural, functional, aesthetic, personal), and critiquing accordingly.

. . . at the "right" time

A number of systems [1, 8] 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 [12, 13] 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. 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 in.

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 *control* critic intervention through 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*-dependenL

Suchman [20] 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 [16] 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 context and possibly to formulate new strategies.

Another approach is suggested by Rittel [15], 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. 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 [21] 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 waterfall models of design according to which designers fffst derive an exhaustive specification of a task and then proceed to methodically implement the specification. Rather, design is viewed as an integrated process of problem framing (task specification), problem solving (design construction), and problem interpretation (interpretive perspectives).

These theoretical considerations suggest that critical reflection is most effective when seen as embedded in a number of inter-dependent contexts. Critiquing mechanisms need to be embedded in design environments in order to support critical reflection in design. Design environments represent a variety of design contexts (see Figure 1). 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 can evolve gradually, supporting design as an argumentative process by incorporating alternative and opposed viewpoints. Second, we represent the current state of *construction* in a graphical display. Third, the evolving partial *specifi cation* is included to guide evaluation of the adequacy of design. Finally, support is provided for the definition of group and personal versions of domain knowledge that can represent critical *interpretations* [18]. By embedding critics in the contexts of the domain, construction, specification, and interpretation, we overcome the problems of stand-alone critic systems.

EMBEDDING CRITICS IN DESIGN ENVIRONMENTS

In response to our evaluation of early critiquing mechanisms and to the theoretical arguments for contextualization, we have explored three types of mechanisms for embedding critics in computational design environments: *generic, specific,* and *interpretive* critics. These mechanisms will be described below in a scenario involving HYDRA [4], a design environment which illustrates our multifaceted architecture.

Integrated Design Environments

Reflection on the shortcomings of JANUS [6] led us to extend it by incorporating representations of additional aspects of the design context. Like its predecessor JANUS, HYDRA contains both a construction and an argumentation component. HYDRA also supports a specification component [9] 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. 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. Perspective mechanisms allow the user to switch viewpoints corresponding to different interests or concerns [18]. 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 argnmentation 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 a specific 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 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. 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.

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 situatiQn-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 1). Bob examines the supporting argumentation, "Having the dishwasher to the left of the sink creates an efficient work flow for a lefthanded 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 rule, "The dishwasher should be on the right side of the sink." Interpretive critics support design as a 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; access to argnmentation concerning designing for resale practices will be provided

Figure 1. This figure shows a screen image of HYDRA. 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 [9]. 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.

Three Embedded Critiquing Mechanisms

Embedded critics increase the usefulness of design environments by making information structures more relevant to the task at hand [9]. 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 possible problem and provides entry into the exact place in the argumentative hypermedia system where the appropriate explanation is located; and (4) concrete catalog examples that illustrate the explanation given in the form of argumentation can optionally be delivered [7].

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 mechanisms for embedded critics differ from one another in how they determine which set of critic rules should be enabled. Generic critics provide the default set of enabled critics by evaluating the construction situation

based on an assumption that a designer wants to design a "good" kitchen. "Good" in this sense refers to a kitchen that meets commonly accepted practices of most kitchen designers.

Specific Critics. Specific critics evaluate the construction situation for compliance with the partial specification. Specification-linking rules are used to dynamically identify the set of specific critics to be enabled [9].

A specification linking rule represents a dependency between an issue/answer pair in the specification and associated pro and con arguments in the argumentation-base. As shown in Figure 2, a specification linking rule connects the argumentation issue "Where should the dishwasher be placed?" with the specification item "Is the primary cook right or left-handed?" The shared domain distinction "left-handedness" is used to establish a dependency between this particular specification item and the argumentation issue.
Figure 2. Illustration of a specification-linking rule that enables the "dishwasher should be on the left side of the sink" critic. The domain distinction associated with a specification item ("left-handedness") is paired with a matching pro or con argument in the argumentation (left-of dishwasher sink) to form a *specific critic rule.*
Each specification item has either an associated critic condition or an associated domain distinction. Domain distinctions are a vocabulary for expressing domain concepts, like left-handedness, safety, and 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.

Interpretive Critics. Interpretive critics [18, 19] 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 are 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.

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 Perspective might include critics and design rationale pertinent to homeowners concerned about their home's resale appeal. Another perspective could be created for the Smith's kitchen; it might include considerations specific to the design of that kitchen.

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, would 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 knowledge inherited from its parents or it can add new knowledge. 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 Figure 3).
Figure 3. Design contexts are arranged in an inheritance network. Three perspectives - the generic, the resale, and the Smith's - are shown. The preferred placement of the dishwasher depends on the perspective selected.
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 which 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.

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 [2, 15, 16] and situated cognition researchers [11, 20, 21].

Recently, we have built design environments in a variety of domains including lunar habitat design [19], phone-based interface design [14], computer network design *[5],* and user interface design [12]. These design environments go beyond conventional CAD systems by modeling domain semantics in several design contexts and not just modeling geometric relationships. Though the knowledge bases of these research prototypes are not exhaustive, they exhibit a high degree of complexity with their many design units, catalog entries, critics, and

domain distinctions. Exploration of these environments has confirmed that simple browsing mechanisms are insufficient and that critiquing mechanisms capable of delivering the right information at the right time are desirable.

Design environments support designers in creating and modifying the problem framing throughout the design process, not just in the beginning. Problem framing in design environments is supported by the specification component, where designers articulate their goals and priorities for the design. The problem framing 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 framing and problem solving *[15]* 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 [16] is signaled to the designer. Resolving the conflict might require a modification of (1) the specification by reframing the problem or (2) the construction by rearranging design units.

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 by being dynamically constructed from domain distinctions tied to specification items. Specific critics reduce the intrusiveness [13] 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 supports the evolution of alternative viewpoints on designs. Using these critics, designers are able to consider their designs critically from multiple viewpoints.

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. Embedded critics focus the attention of the system Qfl 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

We thank the HCC group at the University of Colorado, who contributed to the conceptual framework and the systems discussed in this paper. The research was supported by: the National Science Foundation under grants No. IRI-9015441 and MDR-9253425; the Colorado Advanced Software Institute under grants in 1990/91, 1991/92, 1992/93; US West Advanced Technologies; NYNEX Science and Technology Center, and by Software Research Associates, Inc. (Tokyo).

REFERENCES

1. R. Burton and J. S. Brown, "An Investigation of Computer Coaching for Informal Learning Activites," in *Intelligent Tutoring Systems,* D. Sleeman and 3. S. Brown, Ed., London, Academic Press, 19g2, pp. 79-98.

2. P. Ehn, *Work-Oriented Design of Computer Artifacts,* arbetslivscentrum, Stockholm, 1989.
3. G. Fischer, "A Critic for LISP," *Proceedings of the 10th International Joint Conference on Artificial Intelligence,* Milan, Italy, 1987, pp.177-184.

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

5. *G.* Fischer, 3. Grudin, A. C. Lemke, R. McCall, 3. Ostwald, B. N. Reeves and F. Shipman, "Supporting Indirect, Collaborative Design with Integrated Knowledge-Based Design Environments," *HCI.* Vol. 7 (Special Issue on Computer Supported Cooperative Work), 1992.

6. G. Fischer, A. Lemke, T. Mastaglio and A. Morch, "Using Critics to Empower Users," *CHI '90,* Seattle, WA,

1990, pp.337-347.

7. 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.

8. G. Fischer, A. C. Lemke and T. Schwab, "Knowledge-Based Help Systems," *Human Factors in Computing Systems, CHI'8S Conference Proceedings (San Francisco. CA),* pp.161-167, 1985.

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

10. A. Girgensohn, "End-User Modifiability in Knowledge-Based Design Environments," Technical Report CU-CS-595-92, Department of Computer Science, University of Colorado at Boulder, 1992.

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

12. A. C. Lemke, "Design Environments for High-Functionality Computer Systems," Unpublished Ph.D. Dissertation, Department of Computer Science, University of Colorado at Boulder, 1989.

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

14. A. Repenning and T. Sumner, "Using Agentsheets to Create a Voice Dialog Design Environment," *Symposium on Applied Computing (SAC '92),* Kansas City, MO., 1992, pp.1199-1207.

15. H. Rittel and M. Webber, "Planning Problems are Wicked Problems," in *Developments in Design Methodology,* N. Cross, Ed., John Wiley & Sons, New York, 1984, pp.135-144.

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

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

18. G. Stahl, "Toward a Theory of Hermeneutic Software Design," Technical Report CU-CS-589-92, Computer Science Department, University of Colorado at Boulder, 1992.

19. G. Stahl, "Supporting Interpretation in Design," Accepted to *Journal of Architecture and Planning Research, Special Issue on Computational Representations of Knowledge,* Forthcoming in Summer 1993.

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

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