CONTRIBUTION TO THE ACM CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS (CHI '89)

Topic Area: Interface Design Tools and Techniques (specifically: User Interface Tools Kits; Rapid Prototyping; Intelligent Interfaces and Tutors; Adaptable Systems; Graphics Interfaces)

Design Environments for Constructive and Argumentative Design

Gerhard Fischer and Anders Morch
Department of Computer Science and Institute of Cognitive Science

Raymond McCall
College of Environmental Design and Institute of Cognitive Science

University of Colorado, Boulder, Colorado 80309

Abstract

Design Environments are computer systems which support design by enabling cooperative problem solving between designer and computer. There are two complementary problem solving activities in design: constructive design and argumentative design. We have created two computer-supported environments, CRACK and VIEWPOINTS, to support these two activities.

CRACK is a knowledge-based critic which has knowledge about how kitchen appliances can be assembled into functional kitchens. VIEWPOINTS is a hypertext system based on the IBIS design methodology and contains useful information about the principles of kitchen design. The integration of these two types of systems will eliminate shortcomings of the individual systems.

Number of words in paper: 2987

Keywords: intelligent support systems, design environments, construction kits, human problem-domain communication, critics, hypertext, issue-based information systems, kitchen design

Contact Person: Anders Morch, phone: (303)-492-1592; e-mail: morch@boulder.colorado.edu

1 Introduction

We have developed design environments in the form of computer systems supporting cooperative problem solving between designer and computer[1]. These include hypertext systems and intelligent support systems which are knowledge-based but not expert systems. Our goal is to augment the creative and analytical skills of designers, not to de-skill them by replacing them with automatic design systems.

Our environments support two complementary design activities: *construction* and *argumentation*. Construction is the process of assembling a solution from building blocks appropriate to a particular problem domain. For example, in designing a kitchen the building blocks would include things such as sinks, cabinets, stove, walls and windows. Argumentation is the process of reasoning about a design. It includes both thinking about design principles and discussing a design problem with others. The designer must construct to create any solution, but the quality of the solution depends on the argumentation underlying it.

To support construction we have developed systems which allow assembly of complex artifacts from basic building blocks. These systems incorporate design critics which have knowledge of criteria for fitting the building blocks together to make successful designs. To support argumentation we have developed hypertext systems which aid the creation of argumentation and retrieval of information, including recurring issues, answers and arguments about the problem domain. In the remainder of this paper we discuss our experiences with these systems in the past, two recently constructed systems called CRACK and VIEWPOINTS, and our concepts for integrating the two types of systems (see Figure 1).

2 Background and Motivation

One of the authors (Fischer) has been involved for a number of years in the development of intelligent support systems for design. Evaluation of these systems has shown that their explanation and suggestion facilities--which display brief, canned texts--are insufficient for many purposes. Such texts do not reveal the complex argumentation which underlies them, or the differences of opinion among experts on many design issues. Yet these things can be important information for the designer. In subjective problem domains such as design there is no optimal solution but only a set of alternative solutions which satisfice [Simon 81]. Satisficing is dependent upon a *metric* and a *viewpoint*. An example of a kitchen design metric is the *work triangle*, which connects the center fronts of the sink, stove and refrigerator. One design principle says that the work triangle should be less than 23 feet in perimeter [Jones, Kapple 84]. This allows many alternative kitchen solutions to be satisficing. The work triangle principle can also be interpreted differently by designers with different viewpoints: some allow the work triangle to be up to 26 feet in large kitchens.

Another of the authors (McCall) has been involved for a number of years in development of IBIS (Issue-Based Information Systems) hypertext [Kunz, Rittel 70] for design. In particular, he has developed an approach to IBIS called PHI (Procedural Hierarchy of Issues) [McCall 87] and a hypertext system called MIKROPLIS to support PHI [McCall, Schaab, Schuler 83]. Use of MIKROPLIS over the past five years suggests that its passivity has lead to its being underused. It supports argumentative design, but for most of the design process designers work predominantly in constructive mode. MIKROPLIS has information

[1]By *cooperative problem solving* we mean cooperation between human and computer. This should not be confused with *computer-supported cooperative work* which refers to cooperation among humans. The latter is not discussed in this paper.

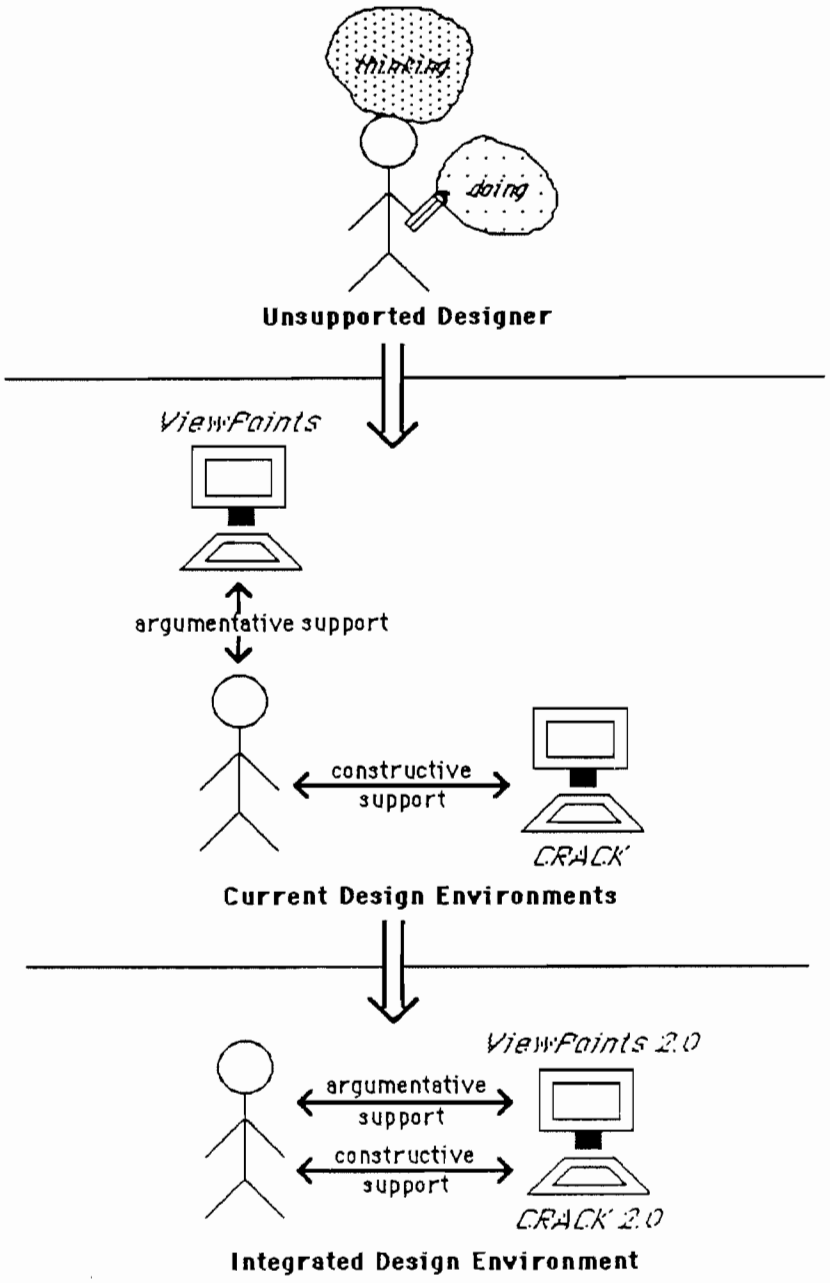

Figure 1: Stages in the Development of an Integrated Design Environment

By observing that designers are involved in two complementary design activities (construction: "doing" design; argumentation: "thinking about" design), we are on the way to develop an Integrated Design Environment which supports designers in these two activities.

useful for construction, and in principle all its information is at least indirectly useful for construction. Nevertheless, students have been reluctant to interrupt construction to look for this information. They often do not know that they need information. If they do, then they do not know if the system contains the information. If it did, they might not know how to retrieve it. To support construction such systems would have to actively alert the designer to the existence of information useful for a task at hand. To do this such systems need embedded active agents. A shortcoming which MIKROPLIS shares with most current hypertext systems is that it is purely passive and contains no such agents [Fischer et al. 88; Halasz 87].

3 CRACK: A Knowledge-Based System for Constructive Design

CRACK is a kitchen design environment [Fischer, Morch 88] consisting of two components: a domain-oriented construction kit for creating a kitchen floor plan layout and a knowledge-based critic for evaluating it.

A *construction kit* comprises a set of domain-specific building blocks. In CRACK these are called design units (DUs). DUs are selected from a palette (see Figure 2), and can be moved around with the mouse to desired locations inside a work area. Useful operations on DUs are `move`, `rotate` and `modify width`. During construction, it is important to give the designer the feeling of directly generating the design without the computer's being "in the way." We refer to this as *human problem-domain communication*: a prerequisite for truly usable design tools.

The goal of human problem-domain communication [Fischer, Lemke 88] is to eliminate computer-specific programming languages and instead to build layers of abstraction with which domain specialists such as kitchen designers feel comfortable. Human problem-domain communication provides a new level of quality in human-computer communication because the important abstract operations and objects in a given area are built directly into the computing environment. The designer can therefore operate with personally meaningful abstractions which result in a reduced transformation distance between problem-oriented and system-oriented descriptions [Hutchins, Hollan, Norman 86]. Protocol studies have shown [Akin 78] that architects use domain related chunks or parts of buildings such as clusters of rooms, individual rooms, areas and furniture when they design. This means that, in kitchen design, designers should be able to select and arrange DUs directly on the screen without any intermediate graphics programming such as defining coordinates or drawing simple shapes.

Construction kits and human problem-domain communication are necessary but not sufficient for useful design environments. Design environments need embedded knowledge for distinguishing "good" designs from "bad" designs and explanations for justifying it. Kitchen design involves more than selecting appliances from a palette; it also involves knowing how to combine these basic building blocks into functional kitchens. Knowledge about kitchen design includes design principles based on building codes, safety standards and functional preferences. An example of a building code is *"the window area shall be at least 10% of the floor area"*; an example of a safety standard is *"the range should not be installed under a window or within 12 inches of a window"* and an example of a functional preference is *"the work triangle should be less than 23 feet"* [Jones, Kapple 84; Paradies 73]. Functional preferences may vary from designer to designer, whereas building codes and safety standards should be violated only in exceptional cases. For the non-expert designer to construct truly useful kitchens, design environments also need knowledge-based *critics*.

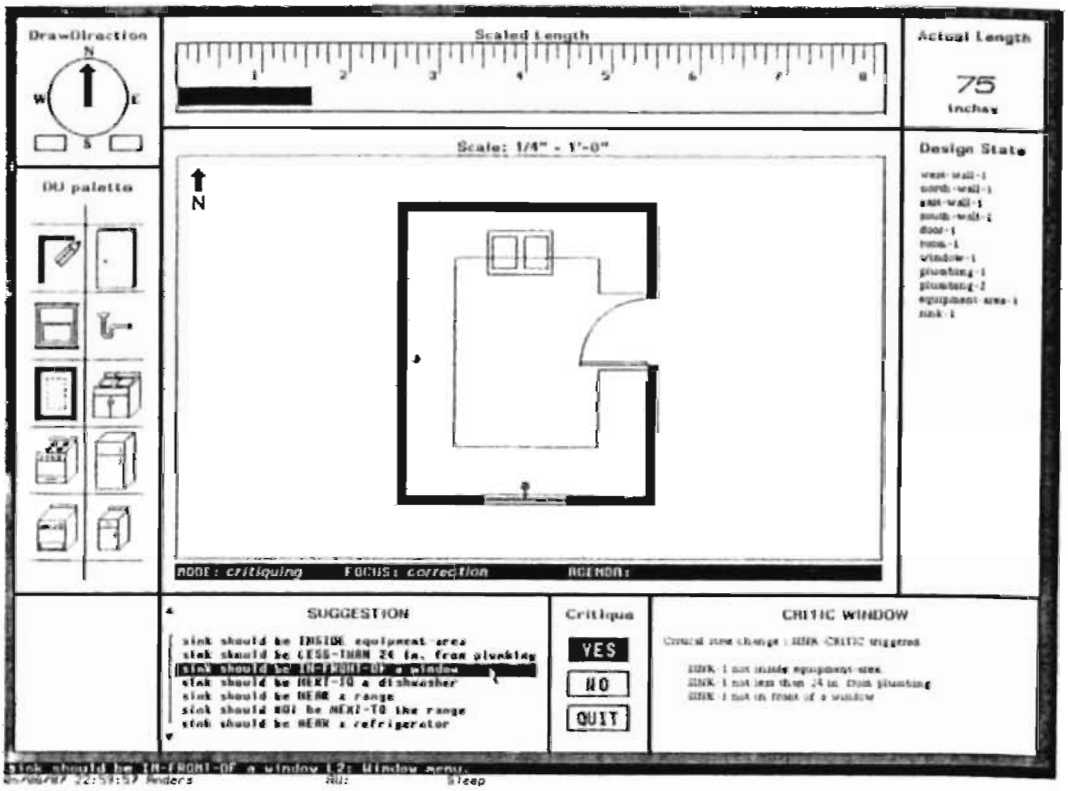

Figure 2: CRACK: Suggestions from the SINK-CRITIC

CRACK's user interface is based on a direct-manipulation interaction style and the metaphor of an "architect's workbench". Operations on building blocks (DUs) are initiated by clicking on their instance name in the Design State window. Suggestions and criticism can be questioned by clicking on mouse sensitive text. Compass, ruler and actual length are active values used during wall drawing and door/window/plumbing positioning to support the designer with graphical data. Critiquing can be turned on and off.

Critics are intelligent support systems which detect and criticize partial solutions constructed by the designer based on knowledge of design principles. Critics in CRACK are state-driven condition-action rules which take action when non-satisficing partial designs are detected (see Figure 3). The critics display criticism (such as: *"sink not in front of a window"*) in a critic window (lower right corner in Figure 2). If a designer wants more information, requests for suggestions (lower left corner in Figure 2) and explanations (lower left corner in Figure 4) can be made by clicking with the mouse on mouse-sensitive text output.

The critics create the text display dynamically and in the proper context as the user is designing. The suggestions and explanations, however, are canned text. Explanations contain justifications for the various kitchen design principles (such as why the sink should be in front of a window). However, simple explanations are often not sufficient to persuade the designer or to capture the argumentation among design experts. For this a system like VIEWPOINTS is needed.

```
;=============================================================
;;; Definition of SINK-CRITIC which test the various relation-
;;; ships defined between sink and other design units.
;=============================================================

(defrule sink-critic
    (mode critiquing)
    (focus correction)
    (enabled sink-critic t)
    (or (moved ?sink)
        (rotated ?sink)
        (scaled ?sink))
    (instance-of ?sink sink)
=>
    ;; Assume first that all relations are violated.

    (bind ?in-front-of-p nil)

    ;; Must first check to see that we have the
    ;; related DUs, if not we don't want to critique.

    (bind ?window (get-schema-value
                    'window 'has-instances))
    (if (not ?window) then
        (bind ?in-front-of-p t))

    ;; Now test each relation for each applicable DU instance.

    (for window inslotvalues 'window 'has-instances do
             (if (in-front-of ?sink window) then
                 (bind ?in-front-of-p t))

    ;; Tell the user which relations are violated

    (if (not ?in-front-of-p) then
        (printout (window-stream 'critic-window)
                  ?sink " not in front of a window.")
        (multiple-value-bind (x y)
            (cursor-x-y-position 'critic-window)
            (create-mousable-area 'in-front-of
                        'critic-window
                        22 y x (+ y 13)
                        "sink not in-front of a window"
                        "Mouse-L: IN-FRONT-OF relation menu."))))
```

Figure 3: Code Example from the SINK-CRITIC

CRACK is written in ART and this example shows part of an ART rule, the SINK-CRITIC. The condition of the rule precedes the => (inference symbol), while the action follows it. This extracted code shows the "in-front-of" test between the sink and a window. There are seven other relational tests like this one in the SINK-CRITIC. "In-front-of" is defined as a method which geometrically checks that at least half of the sink is in front of a window.

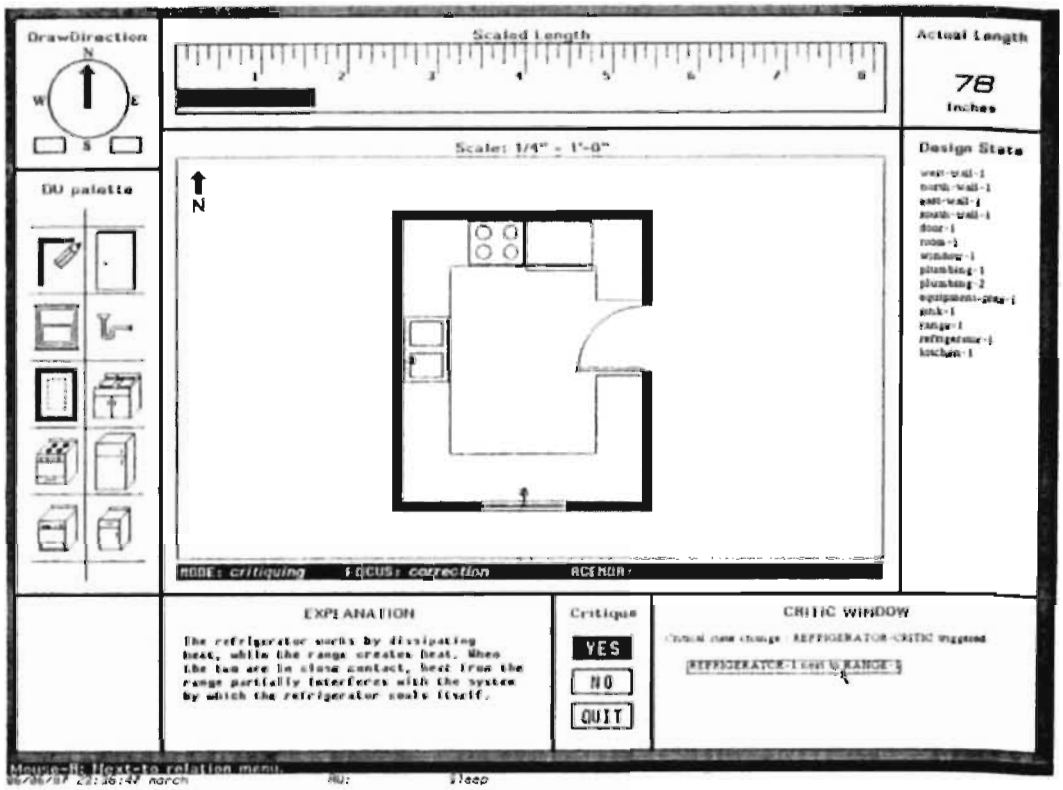

Figure 4: Explanations for NEXT-TO relation in REFRIGERATOR-CRITIC

An explanation for why the refrigerator should not be next to the range. Although the explanation is context sensitive, it is unmodifiable, canned text. This is insufficient for domains where issues are disputed by experts.

4 VIEWPOINTS: A Hypertext System for Argumentative Design

VIEWPOINTS is a hypertext system for argumentative kitchen design based on the PHI design methodology [McCall 87], an extension of IBIS [Kunz, Rittel 70]. VIEWPOINTS is the most recent of a series of PHI and IBIS based systems including MIKROPLIS [McCall, Schaab, Schuler 83] and, more recently, gIBIS [Conklin, Begeman 87]. These are all hypertext systems since the conceptual framework for IBIS fits naturally into the non-linear structuring of text in a network of nodes and labelled links [Conklin 87].

In IBIS design centers on the resolution of issues. In Rittel's original IBIS, issues were to be resolved by the process of deliberation, consisting of (1) considering alternative answers, (2) arguing the pros and cons of these answers and (3) rejecting and accepting answers on the basis of the argumentation. PHI differs from IBIS in two respects: it allows decomposition of issues, answers and arguments into hierarchies of subissues, subanswers and subarguments; and it broadens the concept of *issue* to include all design questions, not merely those deliberated.

VIEWPOINTS extends MIKROPLIS with graphics capabilities, an important feature for illustrating principles of

kitchen design. Currently, VIEWPOINTS is used as a look-up manual where designers can find answers to specific problems and consider the various arguments for and against these answers. VIEWPOINTS is also user extensible; so designers can add argumentation reflecting their own design philosophy.

The elements of VIEWPOINTS are issues, answers, arguments and graphics (see Figure 5). A typical issue is *"What should the location of the sink be?"*. An answer to this could be *"Near a range."*. One argument supporting this answer would be *"There is a frequent work flow from sink to range during food preparation."* Another argument would be *"There should be a minimum of 24 inches counter space between sink and range to allow for 'set-off' space."* The graphics corresponding to this answer could represent pictorially what it means for something to "be near" something else in the domain of kitchen design.

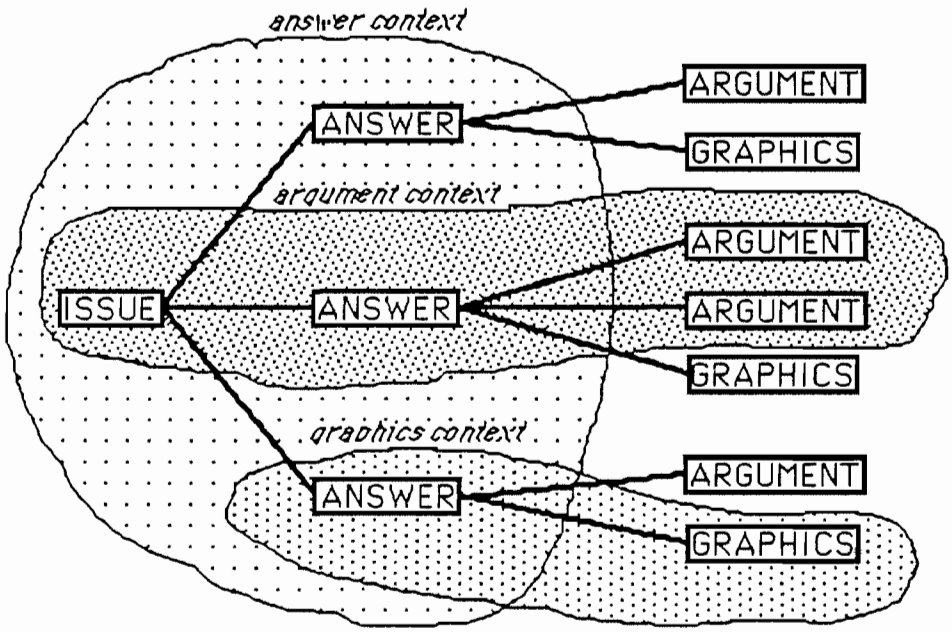

Figure 5: Elements of VIEWPOINTS

VIEWPOINTS consists of issues, answers, arguments and graphics about design problems. Arguments can be either support arguments or counter arguments. Argumentative contexts: argument context, answer context and graphics context define the amount of information presented to the designer on a MACINTOSH[2] screen.

VIEWPOINTS in its current form supplies the designer with an initial set of issues, answers and arguments. This defines a "generic" viewpoint, or the default knowledge of kitchen design principles acquired by interviewing professional designers (see section 6). This initial knowledge is meant to provoke ideas and

[2]VIEWPOINTS has been developed on the MACINTOSH using HYPERCARD.

to define the overall structure of the information base. By having different experienced designers using the system, we allow multiple viewpoints to be incorporated into the information base as each of them adds issues, answers and arguments representing their particular point of interest and expertise.

To fill a screen with as much useful information as possible without overwhelming the designer, we have defined several *argumentative contexts* consisting of subsets of issues, answers, arguments and graphics for simultaneous display. (Figure 5 shows how they are related). The three most useful contexts are:

- Answer context: superissue, issue and answers (Figure 6)
- Argument context: issue, answer and arguments (Figure 7)
- Graphics context: graphics about a particular answer (Figure 8)

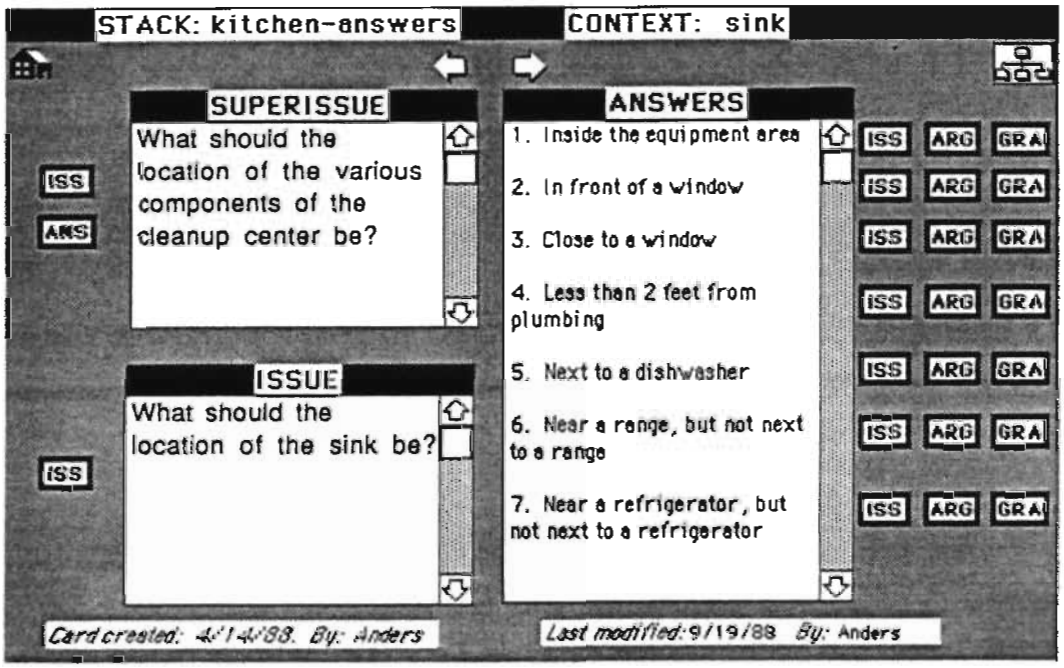

Figure 6: An Answer Card and an Answer Context

This card shows information related to answering the issue "What should the location of the sink be?". The text are typed inside "fields", while the small rectangular areas are "buttons". Buttons link to other cards (ISSue, ANSwer, ARGument, GRAphics) when they receive a mouse click. Cards are stored in stacks. There are four stacks: kitchen-issues, kitchen-answers, kitchen-arguments and kitchen-graphics.

VIEWPOINTS can also be used for systematic browsing through the issue hierarchy. In this case the designer would start from the top-level issue, referred to as the *prime issue*[3], and move stepwise down to more detailed issues.

[3]The prime issue in kitchen design is "What should the design of the kitchen be?"

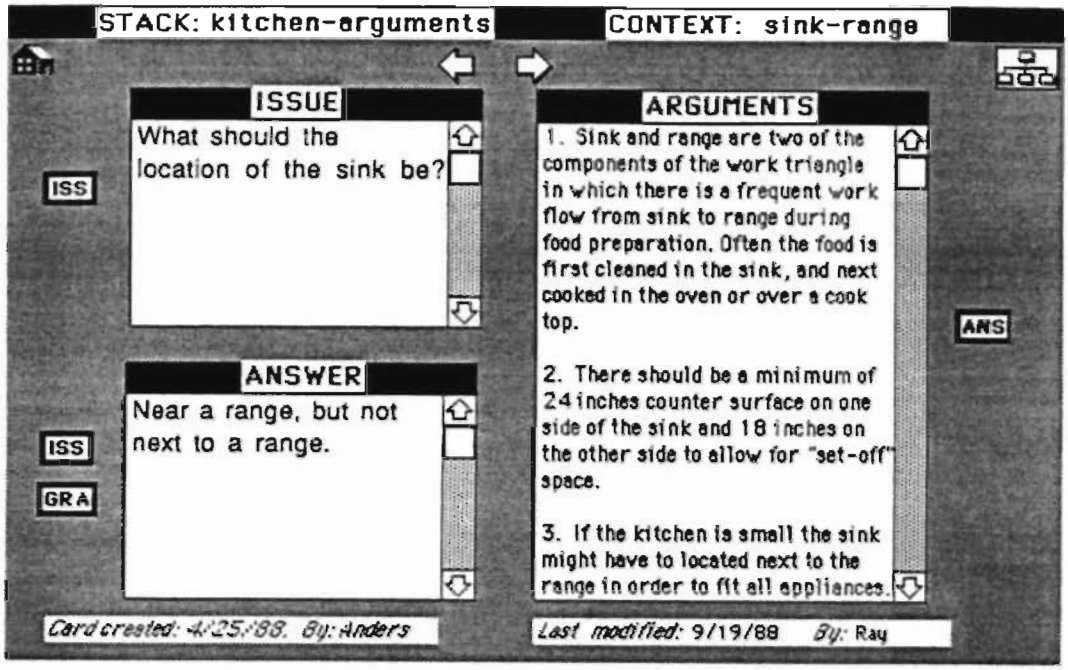

Figure 7: An Argument Card and an Argument Context

5 Empirical Studies and Evaluation

The design knowledge used in CRACK and VIEWPOINTS was acquired through protocol studies of two professional kitchen designers. One designer and several students helped in evaluating CRACK.

Protocol Analysis. The kitchen designers were given scenarios consisting of sample floor plans and hypothetical clients with specific needs. They were asked to think aloud while filling out the room with appliances and cabinets [Ericsson, Simon 84]. These protocol studies revealed domain-specific concepts for kitchen design. Spatial relationships such as *"in front of"*, *"next to"* and *"near"* have their own meaning in this domain. *"In front of"* refers to a relation between a piece of equipment (appliance, cabinet) and a wall fixture (door, window, plumbing), such as "sink *in front of* window", which means that at least half of the sink is in front of the window. *"Next to"* refers to two pieces of equipment which are side by side along a wall, such as "sink *next to* dishwasher". *"Near"* refers to two pieces of equipment which are not immediately next to each other, but still within reach, about 4-8 feet apart, e.g., "sink *near* refrigerator". These relations and others are incorporated in the critics.

Evaluation. To evaluate CRACK, we let computer science students, design students and a kitchen designer use it. This showed that one did not have to be a computer or design expert to use the system. It also suggested that students could produce better designs and learn about design principles through the use of the system [Fischer, Morch 88].

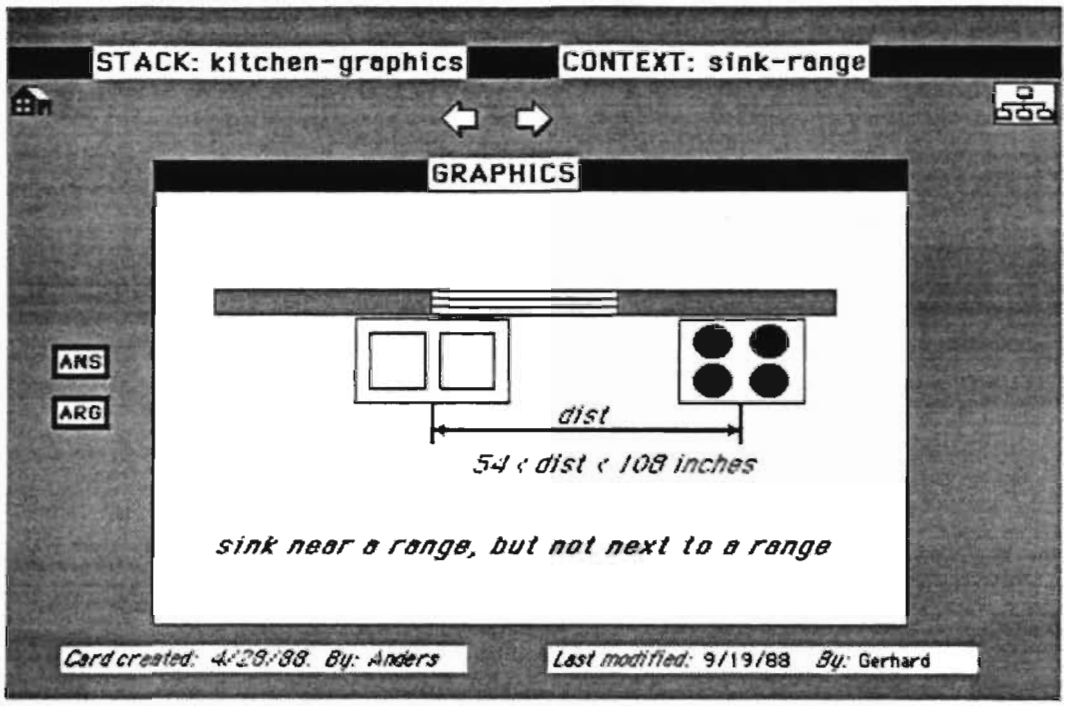

Figure 8: A Graphics Card

Use of VIEWPOINTS revealed that its graphics interface makes it easier to use than the text-only MIKROPLIS system, though the latter has more powerful authoring and retrieval facilities. The ability to include graphic explanations of answers was also a clear improvement. Due to the large number of issues in the information base, we have found that designers often get lost during this extensive browsing since there is no facility available for guiding designers in the right direction[4]. Evaluations of CRACK and VIEWPOINTS indicate that integrated support for construction and argumentation is necessary for full support of design

6 Future Work: An Integrated Design Environment

Designers continuously shift between construction and argumentation [Krauss, Myer 68]. The shift to argumentation occurs when designers need to evaluate what has been constructed or to think about what to do next. The criticism given by CRACK allows this, but in far too limited a way. The triggering of criticism should instead provide an entry point into VIEWPOINTS' far richer information base. Since this triggering is context sensitive, it can be used to identify the issue(s) corresponding to the current construction task. For example, the criticism *"sink not in front of a window"* shows that the designer has problems locating the sink and is therefore implicitly proposing an answer to the issue *"What should the location of the sink be?"*. This fact allows identification of the appropriate "argumentative context" in VIEWPOINTS, including alternative answers, graphics, arguments and subissues. This is a much richer explanation facility than

[4]A way to improve upon this, and another way to integrate AI with hypertext, is to build intelligent agents into VIEWPOINTS which can automatically raise the necessary issues needed to resolve the designer's problem. This is not addressed in this paper.

CRACK currently provides. After studying the argumentation about the design principles relevant to their current problem, designers can return to CRACK and resume construction.

CRACK and VIEWPOINTS currently suffer from being in separate environments (SYMBOLICS and MACINTOSH). Our future work will concentrate on integrating the two within the SYMBOLICS GENERA software environment using CONCORDIA [Walker 88] and the DOCUMENT EXAMINER [Walker 87] as the basis for PHI hypertext. With this integration we expect to eliminate some shortcomings of CRACK and VIEWPOINTS as separate systems.

Acknowledgments

The authors would like to thank our colleagues and students who have helped us critically evaluate the usefulness of CRACK and VIEWPOINTS. We are especially grateful to Sarah Reep and Maggie Boling who as professional kitchen designers have taken time to collaborate with us. The research was supported in part by Grant No. MDA903-86-C0143 from the Army Research Institute and by a grant from NYNEX corporation.

References

[McCall, Schaab, Schuler 83]
R. McCall, B. Schaab, W. Schuler, *An Information Station for the Problem Solver: System Concepts*, in C. Keren, L. Perlmutter (eds.), *Applications of Mini- and Microcomputers in Information, Documentation and Libraries*, Elsevier, New York, 1983.

[Akin 78]O. Akin, *How Do Architects Design?*, in J. Latombe (ed.), *Artificial Intelligence and Pattern Recognition in Computer Aided Design*, North-Holland, New York, 1978.

[Conklin 87]
J. Conklin, *Hypertext: An Introduction and Survey*, IEEE Computer, Vol. 20, No. 9, September 1987.

[Conklin, Begeman 87]
J. Conklin, M. Begeman, *gIBIS: A Hypertext Tool for Team Design Deliberation*, Hypertext'87 Papers, University of North Carolina, Chapel Hill, NC, November 1987, pp. 247-251.

[Ericsson, Simon 84]
K.A. Ericsson, H.A. Simon, *Protocol Analysis: Verbal Reports as Data*, The MIT Press, Cambridge, MA, 1984.

[Fischer et al. 88]
G. Fischer, S.A. Weyer, W.P. Jones, A.C. Kay, W. Kintsch, R.H. Trigg, *A Critical Assessment of Hypertext Systems*, Human Factors in Computing Systems, CHI'88 Conference Proceedings (Washington, D.C.), ACM, New York, May 1988, pp. 223-227.

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

[Fischer, Morch 88]
G. Fischer, A. Morch, *CRACK: A Critiquing Approach to Cooperative Kitchen Design*, Proceedings of the International Conference on Intelligent Tutoring Systems (Montreal, Canada), June 1988, pp. 176-185.

[Halasz 87]
F.G. Halasz, *Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems*, Communications of the ACM, Vol. 31, No. 7, July 1987, pp. 836-852.

[Hutchins, Hollan, Norman 86]
E.L. Hutchins, J.D. Hollan, D.A. Norman, *Direct Manipulation Interfaces*, in D.A. Norman, S.W. Draper (eds.), *User Centered System Design, New Perspectives on Human-Computer Interaction*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1986, pp. 87-124, Ch. 5.

[Jones, Kapple 84]
R.J. Jones, W.H. Kapple, *Kitchen Planning Principles - Equipment - Appliances*, Small Homes Council - Building Research Council, University of Illinois, Urbana-Champaign, IL, 1984.

[Krauss, Myer 68]
R.I.Krauss, *Design: A Case History*, in G.T. Moore (ed.), *Emerging Methods in Environmental Design and Planning*, The MIT Press, Cambridge, MA, 1968.

[Kunz, Rittel 70]
W. Kunz, H. Rittel, *Issues as Elements of Information Systems*, Working Paper 131, Center for Planning and Development Research, University of California, Berkely, CA, 1970.

[McCall 87]
R. McCall, *PHIBIS: Procedurally Hierarchical Issue-Based Information Systems*, Proceedings of the 1987 Conference on Planning and Design in Architecture, American Society of Mechanical Engineers, 1987.

[Paradies 73]
K. Paradies, *The Kitchen Book*, Peter H. Wyden Publisher, New York, NY, 1973.

[Simon 81]
H.A. Simon, *The Sciences of the Artificial*, The MIT Press, Cambridge, MA, 1981.

[Walker 87]
J.H. Walker, *Document Examiner: Delivery Interface for Hypertext Documents*, Hypertext'87 Papers, University of North Carolina, Chapel Hill, NC, November 1987, pp. 307-323.

[Walker 88]
J.H. Walker, *Supporting Document Development with Concordia*, IEEE Computer, January 1988, pp. 48-59.