HUMAN-COMPUTER INTERACTION, 1991, Volume 6, pp. 393-419

# Making Argumentation Serve Design

**Gerhard Fischer, Andreas C. Lemke,** and **Raymond McCall**
*University of Colorado*

**Anders I. Morch**
*NYNEX Science and Technology*

## ABSTRACT

Documenting argumentation (i.e., design rationale) has great potential for serving design. Despite this potential benefit, our analysis of Horst Rittel's and Donald Schön's design theories and of our own experience has shown that there are the following fundamental obstacles to the effective documentation and use of design rationale: (a) A rationale representation scheme must be found that organizes information according to its relevance to the task at hand; (b) computer support is needed to reduce the burden of recording and using rationale; (c) argumentative and constructive design activities must be linked explicitly by integrated design environments; (d) design rationale must be reusable. In this article, we present the evolution of our conceptual frameworks and systems toward integrated design environments; describe a prototype of an integrated design environment, including its underlying architecture; and discuss some current and future work on extending it.

*Authors' present addresses:* Gerhard Fischer, Department of Computer Science and Institute of Cognitive Science, University of Colorado, Boulder, CO 80309-0430; Andreas C. Lemke, GMD-IPSI, Dolivostr. 15, 6100 Darmstadt, Germany; Raymond McCall, College of Environmental Design and Institute of Cognitive Science, University of Colorado, Boulder, CO 80309-0314; Anders I. Morch, NYNEX Science and Technology, 500 Westchester Avenue, White Plains, NY 10604.

**CONTENTS**

## 1. INTRODUCTION

Documenting argumentation (i.e., design rationale) has great potential for improving design. In addition to being invaluable for maintenance, redesign, and reuse, it promotes critical reflection during design. Despite such potential benefits, our experience has shown that there are fundamental obstacles to the effective documentation and use of design rationale. Argumentation does not naturally serve design; it must be made to do so.

The structure of this article follows the history of our work, driven by the development and evaluation of conceptual frameworks and prototype systems. In Section 2, the term *design rationale* is characterized. In Section 3, we discuss issue-based information systems (IBIS) and Procedural Hierarchy of Issues (PHI), two frameworks for representing argumentation. We show that IBIS has fundamental problems. IBIS represents neither dependency relationships between issues nor nondeliberated issues. PHI is a variant of IBIS that remedies these problems. In the past, argumentation has been considered in isolation from the activity of solution construction. The major breakthrough in our thinking, based on observing the shortcomings of the two isolated approaches, was the realization that argumentation must be integrated into the context of construction. In Section 4, we describe approaches to devising tools for construction to reduce the transformation distance from application domain to implementation domain by supporting human problem-domain communication. In Section 5, we discuss *integrated design environments* that unify construction and argumentation. The theoretical basis for this integration is Schön's theory of reflection in action. In Section 6, we describe current and future work on adaptive and reusable domain-oriented issue bases, enriched catalogs, and improved representations of the task at hand.

Throughout the article, we use the JANUS system (Fischer, McCall, & Morch, 1989a, 1989b) as an object to think with. The article discusses aspects of the JANUS system only as they are relevant to our theme; details about JANUS can be found in the references provided.

## 2. DESIGN RATIONALE

***Design.*** To define design rationale, we must first define the term *design*. Similar to design theorists such as Cross (1984), Rittel (1984), Schön (1983), and Simon (1981), we see design not only as problem solving but also as continual problem finding. It is a process of dealing with the kind of "messy situations" that are characterized by uncertainty, conflict, and uniqueness. It is an evolutionary process in which "understanding the problem is identical with solving it" (Rittel, 1972, p. 392), and it can best be characterized by creativity, judgment, and dilemma handling, rather than by objective scientific methods.

We agree with Donald Schön's view of design. For Schön, designing is not primarily a form of problem solving, information processing, or search, but a kind of making. "I shall consider designing as a conversation with the materials of a situation" (Schön, 1983, p. 78). This definition covers a wide range of fields, including architectural (building) design, urban design, software design, hardware design, and various types of engineering design. We call the transactions of designers with materials and artifacts *construction,* which is the activity of creating the actual form of the solution. Construction cannot always be physical but may have to be carried out in the abstract (e.g., on the drafting board). Physical interactions with materials may be too expensive or too dangerous.

***Design Rationale.*** In our approach, design rationale means statements of reasoning underlying the design process that explain, derive, and justify design decisions. A truly complete account of the reasoning relevant to design decisions is neither possible nor desirable. It is not possible because some design decisions and the associated reasoning are made implicitly by construction and are not available to conscious thinking. Some of the rationale must be reconstructed after design decisions have been made. Many design issues are trivial; their resolution is obvious to the competent designer, or the design issue is not very relevant to the overall quality of the designed artifact. Accounting for all reasoning is not desirable because it would divert too many resources from designing itself.

Design rationale in our approach is a synonym for argumentation. Rittel was the first to advocate systematic documentation of design rationale as part of design (Rittel, 1972). He sees design problems as fundamentally open ended and controversial in the sense that there are no objective criteria for closing problem definitions and settling disagreements. Such closing and settling are necessary for design, but, for the designer, the decisions on closing and settling are judgmental and political in nature. The design rationale takes the form of a network of issues (design questions), selected and rejected answers, and arguments for and against these answers (see Section 3).

***The Promise of Design Rationale.*** Design rationale serves design if it helps designers (a) to improve their own work, (b) to cooperate with other people holding stakes in the design, and (c) to understand existing artifacts (i.e., communicate with past designers). Design rationale can trigger critical thought in the individual designer. Writing an idea down allows the designer to make the transition from simply creating that idea to thinking about it.

Design rationale can serve as a memory aid not only to individuals but also to groups (Conklin & Begeman, 1988) by providing a forum for airing issues crucial for coordinating group activities. It is useful for triggering and focusing discussion among members of a project team. By making the processes of reasoning public, it extends the number of people who can participate in the critical reflection of decisions. This reduces the chances of missing some important consideration, and it rationalizes discussion.

To alter a design sensibly—adding, fixing, or modifying features—it is crucial to have an understanding of why it has been designed the way it has. Without knowing the rationale, a designer is apt to violate constraints and to repeat errors by ignoring what previous designers have learned. That is, the rationale created in one design project may be a resource for future, related design projects. Even if the difficulties encountered in a project are not overcome, they might still be informative for future designers. The mere existence of unforeseen problems is itself valuable information. Often, design is based on mistaken predictions of how the artifact will perform in use. If these predictions are documented, they can be compared to actual use. This allows for the development of better theories for predicting performance.

## 3. SUPPORT FOR ARGUMENTATION

On the basis of his theory of wicked problems, Rittel (1984) rejected the efforts by the majority of design methodologists to automate design reasoning. The argumentative approach tries to enhance design by improving the reasoning underlying it and is aimed at supporting the reasoning of human designers rather than replacing it with automated reasoning processes (Fischer, 1990; Stefik, 1986).

## 3.1. IBIS

IBIS (see Kunz & Rittel, 1970) is a method (not a computer system) for structuring and documenting design rationale. The central activity of IBIS is deliberation, that is, considering the pros and cons of alternative answers to questions. The questions deliberated are called *issues*. Proposed answers—including ones that are mutually exclusive—are called *answers* or *positions*. Statements of the pros and cons of answers are called *arguments*. The decision as to which answers to accept and reject is called the *resolution of the issue*.

The various issue deliberations are connected by a variety of interissue relationships. The original IBIS included "more general than," "similar to," "replaces," "temporal successor of," "logical successor of," and others. Graph diagrams with labeled notes and links representing issues and their relationships were used for visualization. Such diagrams, called *issue maps*, were meant to facilitate navigation through the IBIS "problemscape."

From 1970 to 1980, a variety of projects was undertaken that attempted to use IBIS in real-world settings. These projects included IBIS systems for the United Nations, the Commission of European Communities, the (West) German Parliament, the German Federal Office of the Environment, and the German Office of Health (Reuter & Werner, 1983). None of these systems got past the pilot project stage. At the end of this stage, each was judged as somehow failing to serve adequately the design tasks for which it had been created.

After a decade of intensive and generally well-funded efforts to implement IBIS, it became difficult to believe that the failures to do so were coincidental. Clearly, there were fundamental problems with the IBIS method or the approach to implementing it.

The identification and solution of fundamental problems in the creation and use of issue-based design rationale have been central concerns of our research. The first step in this research was a critique of IBIS and an improved issue-based method called PHI (McCall, 1979). The next step was the proposal of a new sort of software technology, hypertext, to handle issue-based rationale (McCall, Mistrik, & Schuler, 1981). We next look at these suggested improvements and the results of their implementation and use.

## 3.2. PHI and the Critique of IBIS

McCall (1978/1979) suggested that there are two related types of information that are omitted from IBIS but that are required for an issue-based approach to serve design effectively. The first and most basic is dependency relationships between issue resolutions, that is, relationships representing the fact that the answering of issues often depends on how other issues are answered. IBIS has no way of representing such dependencies; instead, it treats issue-resolution processes as if they were separable.

The second type of information omitted from IBIS is questions that are not deliberated, that is, questions for which pros and cons of alternative answers are not considered. IBIS ignores these in favor of those questions with which debate and controversy are most likely to be associated. Yet nondeliberated questions occur frequently in design and can influence the resolutions of issues. Furthermore, many such questions themselves have answers that depend on the resolutions of issues.

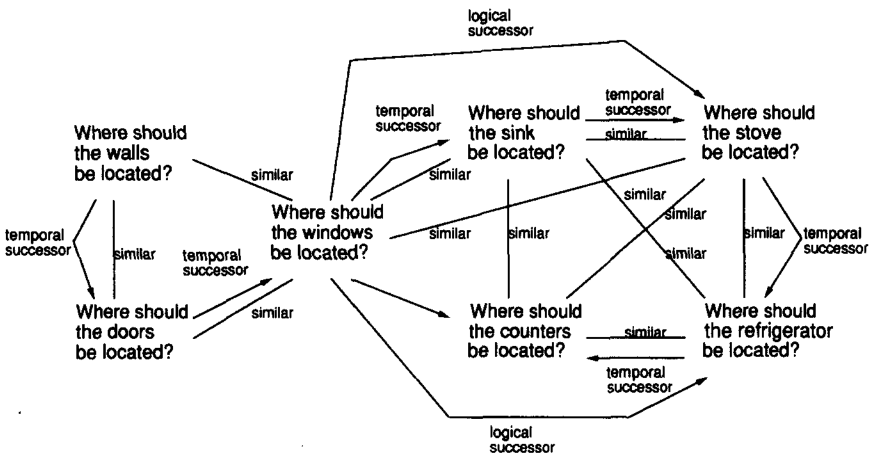
In an effort to overcome these limitations of IBIS, McCall (1991) developed the PHI approach to documenting design rationale. PHI, like IBIS, is a design method rather than a piece of software. It differs from IBIS in two crucial respects: It uses a broader definition of the concept *issue,* and it uses a new principle for linking issues together.

In IBIS, the term *issue* denotes a design question that is deliberated; in PHI, however, every design question counts as an issue, whether deliberated or not. PHI also abandons the interissue relationships proposed by Rittel—"temporal successor of," "similar to," "replaces," and so on. Instead, it uses *serve* relationships. We say Issue A serves Issue B if and only if the resolution of A influences the resolution of B. The dominant type of serve relationship used in PHI is the "subissue of" relationship, which indicates that resolving one issue is a subtask of the task of resolving another. More formally, we say Issue A is a subissue of Issue B if and only if A serves B and B is raised before A. Note that this means that A's being a subissue of B implies that A serves B, and A's serving B does not in itself imply that A is a subissue of B.

In Rittel's IBIS, as evidenced by the many years of real-world and student projects, an issue map is characteristically a dense and tangled network of issues connected by a half dozen different relationships (see Figure 1). In PHI, however, an issue map is a simple quasi-hierarchical structure connected only by serve relationships and having a single root issue (see Figure 2). This structure is tree-like but is seldom a pure tree, because issues can share subissues (Figure 2). The root of a PHI issue map is an issue that represents the project as a whole. For example, if one is designing a kitchen, the root issue might be, "What should be the design of this kitchen?"

PHI has been in nearly continual test use both with and without computer support since 1977. This testing has been informal rather than in the framework of a formal, experimental setting. Furthermore, the testing has emphasized intensive use by relatively few users at a time rather than extensive use by many people—a style we have found to be especially informative for system-building efforts. Testing began in 1977 and 1978 with students at the University of California, Berkeley (McCall, 1978/1979). It

*Figure 1.* **An IBIS issue map.**


logical successor
temporal successor
Where should the sink be located?
temporal successor
similar
Where should the stove be located?
Where should the walls be located?
similar
similar
Where should the windows be located?
temporal successor
similar
temporal successor
similar
similar
similar
similar
similar
similar
temporal successor
Where should the doors be located?
similar
Where should the counters be located?
similar
Where should the refrigerator be located?
temporal successor
logical successor

continued in Heidelberg, Germany, from 1979 to 1984. Since 1984, continual test use has been made of PHI at the University of Colorado, Boulder.

Testing in Berkeley used eight undergraduates and was spread out over a 2-year period. The most important results of this were the generation of issue bases (i.e., networks of issues) that showed the applicability of the PHI to student design projects.

## 3.3. PHI Hypertext

In Heidelberg, testing of PHI began with the attempt in 1979 to use PHI with only typewriters and word processors. By 1980, these efforts ran into severe difficulties in managing the issue-base information. In particular, the information management tasks were so labor intensive and error prone that the decision was made to attempt to develop computer support for PHI. The system developed, called MIKROPLIS (McCall, Mistrik, & Schuler, 1981), became the first issue-based hypertext system.

The defining characteristics of hypertext are nonlinear structure and navigation. The need for the former was understood at the beginning of the MIKROPLIS project (McCall, 1978/1979). The need for the latter emerged in 1982 from working with early users of MIKROPLIS who repeatedly pointed to displayed nodes and asked how to retrieve the nodes linked to them.

Since the beginning of the MIKROPLIS project, several other issue-based hypertext systems have been developed. These include Rittel's own system (Conklin, 1987), gIBIS (Conklin & Begeman, 1988), JANUS-ARGUMENTATION

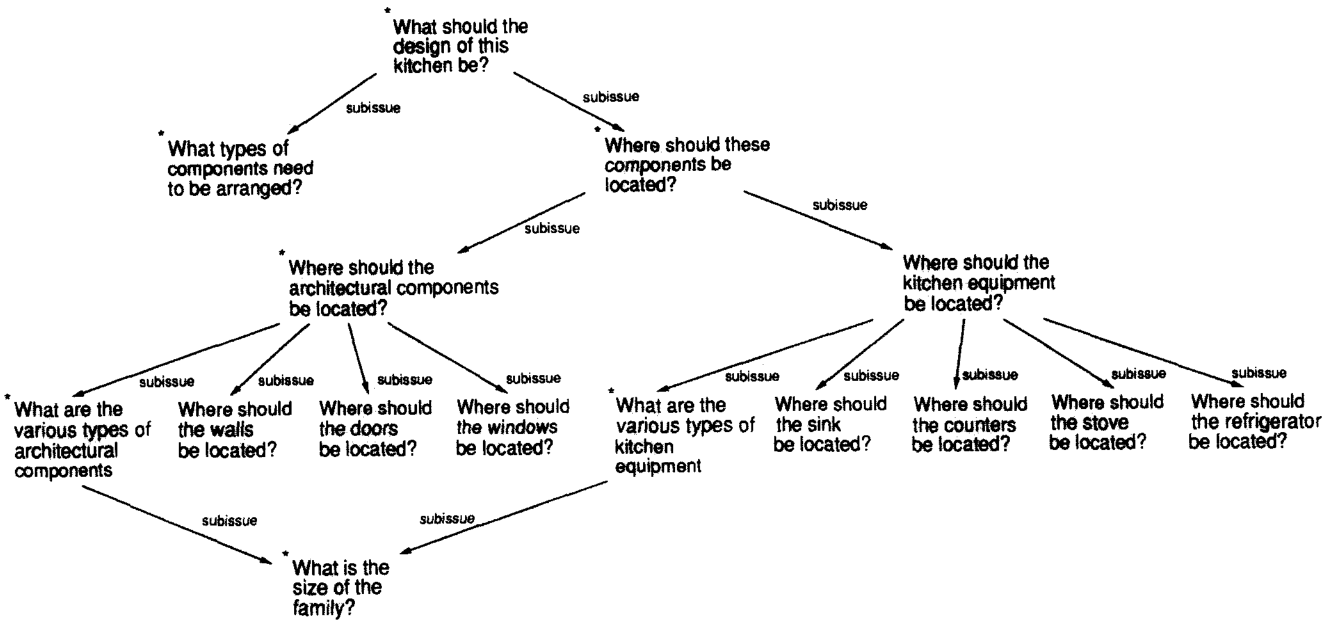

*What should the design of this kitchen be?
subissue
subissue
*What types of components need to be arranged?
*Where should these components be located?
subissue
subissue
*Where should the architectural components be located?
Where should the kitchen equipment be located?
subissue
subissue
subissue
subissue
subissue
subissue
subissue
subissue
subissue
*What are the various types of architectural components
Where should the walls be located?
Where should the doors be located?
Where should the windows be located?
*What are the various types of kitchen equipment
Where should the sink be located?
Where should the counters be located?
Where should the stove be located?
Where should the refrigerator be located?
subissue
subissue
*What is the size of the family?

*Figure 2.* **A PHI issue map. The starred issues, which are not deliberated, are dealt with by PHI but not by IBIS.**

(Fischer et al., 1989a), and PHIDIAS (McCall et al., 1990). MIKROPLIS, PHIDIAS, and JANUS-ARGUMENTATION differ from the others by using PHI rather than IBIS.

To further test PHI and the computer support being developed for it, the MIKROPLIS team kept a PHI issue base for the design of the system. As soon as MIKROPLIS became usable, this issue base was maintained using the MIKROPLIS system itself. This self-referential use encouraged a certain level of awareness and honesty about the performance of PHI and MIKROPLIS.

Additional testing of PHI and MIKROPLIS involved the development of issue bases with MIKROPLIS by a dozen users of various kinds over a period of 3 years. These users included MIKROPLIS project members, people from other project groups within the organization in Heidelberg, and several "knowledge workers" from other organizations. In 1984, an American physician was hired to test the system on a full-time basis for 3 months by attempting to develop an issue base on health care policy. In 11 weeks, he developed a tightly structured issue base equivalent to exactly 500 single-spaced pages in length. This was taken by the physician and others as evidence of the usefulness and usability of both PHI and MIKROPLIS. In particular, the physician felt that he could not have achieved these results with alternative methods or technologies.

Despite this success, there were still problems with using both PHI and MIKROPLIS. The artifact the physician was trying to produce was the issue base itself. To those for whom the issue bases were only means for designing other kinds of artifacts, the use of PHI involved a great deal of work over and above the ordinary work of design. MIKROPLIS substantially reduced the errors and secretarial work of creating an issue base, but there remained a large amount of conceptual and editorial work. Many people were therefore disinclined to use PHI because the costs of invested effort exceeded the immediate payoff. For them, even with MIKROPLIS support, PHI still did not serve sufficiently the design task at hand.

## 3.4. Grounding Argumentation in Construction

PHI hypertext with domain-oriented issue bases reduced the cost and increased the benefits of design rationale. But as our systems dealt successfully with this aspect of design rationale, another, more fundamental obstacle was revealed. There is a crucial design activity not supported by argumentative hypertext: *construction*. In fields such as architectural design, construction is a graphic activity traditionally done by drawing. Construction is the *sine qua non* of design, for no design project can be completed until the construction is done. Argumentation gets its usefulness in design only by influencing construction. For argumentation to serve design, it must serve construction.

Test use of PHI at the University of Colorado, Boulder, provided evidence for the need to integrate argumentation with construction. The test use began with two junior-level undergraduate environmental design studios, each with about 20 students taught by Raymond McCall in 1985. Each studio involved the same semester project: designing a neighborhood shopping center at a particular location in Boulder. Students were asked to record their rationale in PHI form during the project. In both studios, this worked well until students began working out the details of the solution form, that is, actual drawings of buildings. At this point, it became effectively impossible to get students to document their rationale.

To see if these difficulties were independent of instructor and project, two independent study students were asked to document a studio on housing design taught by a nationally known architect. In an effort to keep this inquiry unbiased, the students who did the documentation were not told anything about the hypothesis being investigated and were given only minimal supervision by McCall. The student produced a 175-page document in PHI form, representing the work of a project group of five students in the studio. Again, the documentation of rationale ceased shortly after the construction of solution form began. According to the students who did the documentation, the project group members became unable or unwilling to talk to the documenters as form generation began.

The difficulties encountered in attempting to document the studio projects suggested that there was a fundamental incompatibility between form construction and PHI. To understand what this incompatibility might be, McCall made a series of three videotaped think-aloud protocols of student designers from the College of Environmental Design.

The first protocol involved two juniors who worked for 6 weeks on the design of a store. The second involved a senior working for 10 weeks on the design of a house. The third involved a single senior working for 3 weeks on the design of a kitchen. All were analyzed informally and the second was selected for intensive formal analysis. In particular, representative sections of the form construction process were transcribed and compared to the structures of PHI on a sentence-by-sentence and drawing-by-drawing basis. These results suggested some revisions of PHI (e.g., more explicit representation of criteria and better representation of hypothetical reasoning). On the whole, however, there was a clear match between the processes the student used in form generation and the processes represented in PHI.

The student who created the protocol was asked whether he felt the conclusions of this analysis were accurate. Before being shown the actual videotapes of his protocol, he claimed that he would not be able to think in PHI form while he was designing. When shown the videotapes and their analysis, he agreed that the analysis was correct but professed great surprise at this fact.

At first, these results were quite puzzling. It seemed that students claimed not to be able to use exactly the kind of thinking that they in fact used. Eventually, we found a solution to this puzzle in Schön's theory of reflection in action (Schön, 1983), which is explained later. This theory suggests that the problem was not that students could not think in a PHI-type manner while they devised a solution form but, rather, that they could not be self-consciously aware of doing so. The principle is the same as that which makes it impossible to watch one's own fingers while playing the piano and which incapacitated the fabled centipede who attempted to think about his feet while running.

In the past 3 years, additional informal testing of PHI has gone on at Boulder, Colorado, within the framework of an undergraduate course on design theory and methods. Each of the three times this course has been offered, a consistent pattern has emerged that confirms the earlier results: Students do not deal with issues of form construction until given a project that requires them to do so. To do this project, students rely heavily on taped protocols.

One reason for the need to support construction is that design argumentation is densely populated with deictic references to parts of the partially constructed solution. Without the ability to relate construction and argumentation to each other, it is impossible to discuss the solution. Without construction situations, design rationale cannot be contextualized. Students using our systems to generate issue-based design rationale invariably left out all the issues dealing with construction. They instead concentrated on philosophical discussion, requirements, programmatic analysis, and other preparatory issues rather than actually getting into the design.

Another problem was that serve relationships were often not effective in helping the designer to generate the important rationale. Designers tended to waste time on issues with little impact on the outcome of the project. This too resulted from lack of support for construction. Designers were often unable to judge the relative merits of issues because they could not see their influence on construction. It is only by being relevant to construction that issues serve the project. The serve relationships of PHI showed that resolving one issue was valuable for resolving another. They could not, however, guarantee that any issue served the project as a whole, for this depended on its influencing construction. This lack of relevance to construction promoted what architects call "talkitecture" (i.e., extended discussion having little impact on the solution).

In a good design project, construction generates and regulates argumentation. Argumentation arises out of construction and is often tested by construction. Creating good design rationale requires support for construction.

## 4. SUPPORT FOR CONSTRUCTION

Construction, a subactivity of design, is the composition of elementary building blocks or materials to form an artifact. Sometimes the designer constructs the artifact directly, but in many domains the designer constructs it by making a model or plan of the artifact to be realized by others. The elementary building blocks and materials available for construction activities form the *design substrate.*

Construction is a crucial aspect of design because it creates situations that can "talk back" to the designer:

> Typically [the designer's] making process is complex. There are more variables—kinds of possible moves, norms, and interrelationships of these—than can be represented in a finite model. Because of this complexity, the designer's moves tend, happily or unhappily, to produce consequences other than those intended. When this happens, the designer may take account of the unintended changes he has made in the situation by forming new appreciations and understandings and by making new moves. He shapes the situation, in accordance with his initial appreciation of it, the situation "talks back," and he responds to the situation's back-talk. (Schön, 1983, p. 79)

***Human Problem-Domain Communication.*** The substrate used to design computer-based artifacts typically consists of low-level abstractions (e.g., statements and data structures in programming languages and primitive geometric objects in engineering computer-aided design). Abstracts at that level are far removed from the concepts that form the basis of thinking in the application domains in which these artifacts are to operate. The great transformation distance between the design substrate and the application domain (Hutchins, Hollan, & Norman, 1986) is a reason for the high cost and the great effort necessary to construct artifacts using computers. To reduce this transformation distance, high-level, domain-oriented substrates are required. Akin (1978) and others have shown that designers design with meaningful abstractions at different levels. For example, architects use domain-related chunks or parts of buildings such as clusters of rooms, individual rooms, areas, and furniture when they design.

Rather than communicating with computers, designers should perceive design as communication with an application domain; the computer should become effectively invisible. Human problem-domain communication (Fischer & Lemke, 1988) tries to achieve this goal. It 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. In an environment supporting human problem-domain communication, designers build artifacts from application-oriented building blocks according to the principles of that domain, not the principles of software or geometry.

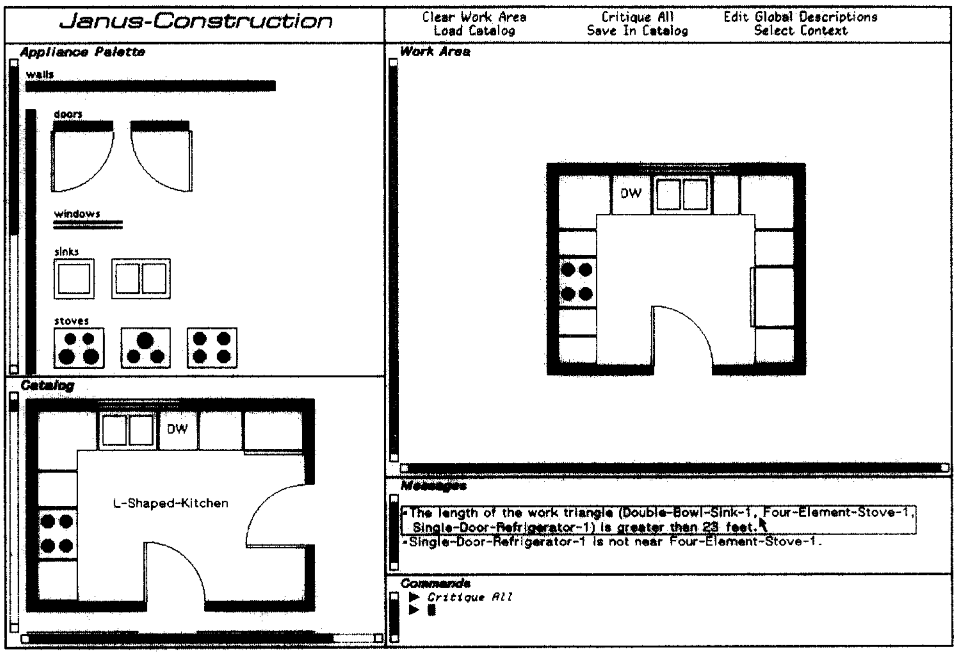
***Construction Kits.*** These kits (Fischer & Lemke, 1988) support human problem-domain communication by offering domain-oriented building blocks presented in a palette and a work area for construction by direct manipulation. Interacting with a computer-based construction kit does not provide the same back-talk afforded by designing with real objects. However, construction kits are an active medium that can react to the designer's actions in ways that are different from real objects. To illustrate the concept of a construction kit, we describe JANUS-CONSTRUCTION, a part of the JANUS system for the domain of residential kitchen design.

JANUS-CONSTRUCTION is a construction kit for the domain of kitchen design. The palette of the construction kit contains domain-oriented building blocks called *design units,* such as *sink, stove,* and *refrigerator* (Figure 3). Designers construct by obtaining design units from the palette and placing them into the work area. They can thus see how different configurations fit the floor plan and how requirements about storage space, work flow, and other considerations can be satisfied—A situation is constructed that can talk back to a skilled designer.

In addition to design by composition (using the palette and constructing an artifact from scratch), JANUS-CONSTRUCTION also supports design by modification. Existing designs can be modified by retrieving them from the catalog and manipulating them in the work area. The catalog can also serve as a learning tool. The user can copy both good and bad examples into the work area. The system can critique such designs to show how they can be improved, thus allowing users to learn from negative examples. Designers can learn about the good features of prestored designs as well.

Designers using JANUS-CONSTRUCTION expressed a sense of accomplishment in using the system because it enabled them to construct something quickly without having detailed knowledge about computers. But construction kits do not in themselves lead to the production of interesting artifacts (Fischer & Lemke, 1988; Norman, 1986). Construction kits do not help designers perceive the shortcomings of an artifact they are constructing. In that they are passive representatives, constructions in the work area do not talk back unless the designer has the skill and experience to form new appreciations and understandings when constructing. Designers often do not see characteristics that lead to breakdowns in later use situations. As Rittel put it: "Buildings do not speak for themselves." Designers who are unaware of the work triangle rule do not perceive a breakdown if that rule is violated (i.e., if the total distance between stove, sink, and refrigerator is greater than about 23 ft).

***Figure 3.*** **JANUS-CONSTRUCTION: the work triangle critic. JANUS-CONSTRUCTION is the construction part of JANUS. Building blocks (design units) are selected from the palette and are moved to desired locations inside the work area. Designers can reuse and redesign complete floor plans from the catalog. The messages pane displays critic messages automatically after each design change that triggers a critic. Clicking with the mouse on a message activiates JANUS-ARGUMENTATION and displays the argumentation related to that message (see Figure 5).**



•The length of the work triangle (Double-Bowl-Sink-1, Four-Element-Stove-1, Single-Door-Refrigerator-1) is greater than 23 feet.
•Single-Door-Refrigerator-1 is not near Four-Element-Stove-1.

▶ Critique All

***Critics.*** Critics operationalize Schön's (1983) concept of a situation that talks back. They use knowledge of design principles to detect and critique suboptimal solutions constructed by the designer.

The critics in JANUS-CONSTRUCTION identify potential problems in the artifact being designed. Their 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 stove should be at least 12 in. away from a door"; and an example of a functional preference is the work triangle rule (Jones & Kapple, 1984; Paradies, 1973). Functional preferences may vary from designer to designer, whereas building codes and safety standards should be violated only in exceptional cases.

Critics detect and critique partial solutions constructed by the designer based on knowledge of design principles. Critics' knowledge is represented as relationships between design units. The stove design unit, for example, has critics with the following relations: **away-from stove door, away-from stove window, near stove sink, near stove refrigerator,** and **not-immediately-next-to stove refrigerator.** These critics are implemented as condition-action rules, which are tested whenever the design is changed. The changes that trigger a critic are operations that modify the design in the work area. When a design principle is violated, a critic will fire and display a critique in the messages pane of Figure 3. In the figure, the work triangle critic fired telling the designer that the "work triangle is greater than 23 feet." This identifies a possibly problematic situation (a breakdown) and prompts the designer to reflect on it. The designer has broken a rule of functional preference, perhaps out of ignorance or by a temporary oversight.

Users can modify and extend JANUS-CONSTRUCTION by modifying or adding design units, critic rules, and relationships (Fischer & Girgensohn, 1990). This end-user modifiability allows for evolution of the environment as design practice and requirements change. Designers can also modify critic rules when they disagree with the critique given. Standard building codes (hard rules) should not be changed, but functional preferences (soft rules) vary from designer to designer and, thus, can and should be adapted. Designers have the capability to express their preferences. For example, if designers disagree with the design principle that the stove should be away from a door, they can edit the stove-door rule by replacing the **away-from** relation between stove and door with another relation (selected from a menu) such as **near.** After this modification, they will not be critiqued when a stove is not away from a door.

***Lack of Argumentative Support.*** The advantage of constructing something is that the constructed artifacts and situations can talk back to the designer. The back-talk of the situation is enriched in our framework with the critics, but the short messages the critics present to designers cannot reflect the complex reasoning behind the corresponding design issues. To overcome this shortcoming, we initially developed a static explanation component for the critic messages (Lemke & Fischer, 1990; Neches, Swartout, & Moore, 1985). The design of this component was based on the assumption that there is a "right" answer to a problem, but the explanation component proved to be unable to account for the deliberative nature of design problems. Therefore, argumentation about issues raised by critics must be supported, and argumentation must be integrated into the context of construction.

## 5. INTEGRATED DESIGN ENVIRONMENTS

Separate systems for construction and argumentation have major deficiencies (as articulated in the previous sections and by Fischer et al., 1989a). If argumentation is to serve design, it must do so by informing construction. If construction acknowledges the nature of design processes (messy situations that are characterized by uncertainty, conflict, and uniqueness), it must have access to the argumentative component. This can happen only if construction and argumentation are explicitly linked in an integrated design environment.

## 5.1. Reflection in Action

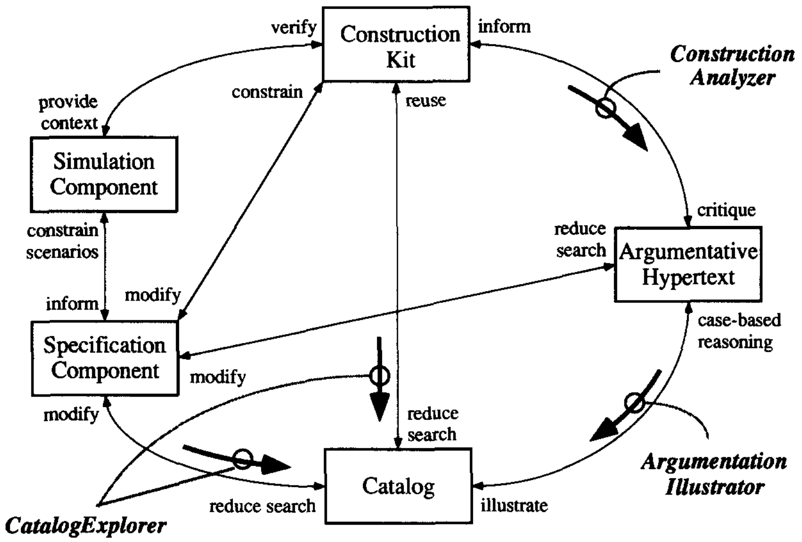
Our original attempt at integrating construction and argumentation was to have construction take place within the framework of argumentation—in other words, to raise an issue for each construction step ("What should the next step be?"), deliberate it, and turn the resolution into a constructive action. Unfortunately, trials of this approach with design students showed that it did not work (see Section 3). A reason for this failure can be found in Schön's theory of design. Schön portrayed design as a continual alternation between two radically different and mutually exclusive types of design processes: knowing in action and reflection in action.

> In a good process of design, this conversation with the situation is reflective. In answer to the situation's back-talk, the designer reflects-in-action on the construction of the problem, the strategies of action, or the model of the phenomena, which have been implicit in his moves. (Schön, 1983, p. 79)

*Knowing in action* is the unself-conscious, nonreflective doing that controls the situated action of constructing the actual artifact. *Reflection in action* is the self-conscious, rational process of reflecting about this action within the "action present," that is, the time period during which reflection can still make a difference to what action is taken. Reflection is required when there is a breakdown in knowing in action. Such a breakdown typically occurs when action produces unforeseen consequences, either good or bad. When a breakdown occurs, reflection can be used to repair the breakdown situation, and then action can continue.

Schön's concepts do not in themselves tell us what the architecture of design support environments should be. His concepts must be operationalized further and augmented substantially if they are to provide a basis for computer-based systems. In our work, we interpret *action* as "construction" and *reflection* as "argumentation." For argumentation to get used, it must be part of reflection in action, implying that it should be brought to the designer's attention only in breakdown situations. Construction cannot be done within an argumentative framework because the former implies unself-conscious, nonreflective engagement in creating the solution, whereas the latter implies self-conscious, reflective thinking about the solution. Argumentation must take place within the action present, that is, within the time period during which it can still make a difference to what action is taken. If the time

***Figure 4.*** **A multifaceted architecture. The links between the components are crucial for exploiting the synergy of the integration.**



required to read and/or record the argumentation is greater than the action present, design is disrupted and the required context is lost. Design rationale can aid reflection by informing with design knowledge, principles, and ideas and by triggering critical thought in the designer. Schön's theory, when operationalized, can then be used as the basis for a system architecture.

## 5.2. An Architecture for Integrated Design Environments

Over the last few years, we have developed an integrated, multifaceted architecture for design environments (see Figure 4). The multifaceted architecture consists of the following five components:

- A construction kit is the principal medium for implementing design. It provides a palette of domain-specific building blocks and supports the construction of artifacts using direct manipulation and form filling.
- An argumentative hypertext system contains issues, answers, and arguments about the design domain. Users can annotate and add argumentation as it emerges during design processes.
- A catalog provides a collection of prestored design examples illus-

trating the space of possible designs in the domain and supporting reuse and case-based reasoning.

- A specification component allows designers to describe some 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 articulated fully at the beginning. They are used to retrieve design objects from the catalog and to filter information in the hypertext.
- A simulation component allows designers to carry out "what-if" games simulating usage scenarios with the artifact being designed.

***Integration.*** The multifaceted architecture derives its essential value from the integration of its components and the links between the components. Used individually, the components are unable to achieve their full potential. Used in combination, however, each component augments the value of the others, forming a synergistic whole. At each stage in the design process, the partial design embedded in the design environment serves as a stimulus to users for suggesting what they should attend to next.

Links among the components of the architecture are supported by various mechanisms (see Figure 4):

- *CONSTRUCTION ANALYZER:* Users need support for construction, argumentation, and perceiving breakdowns. Experience with our early systems has shown that users too often fail to hear the situation talk back; breakdowns do not occur that trigger reflection in action. Additional system components are needed to signal breakdowns. This is the role of the CONSTRUCTION ANALYZER in the multifaceted architecture. The CONSTRUCTION ANALYZER is a version of the critics described in Section 4 enhanced with pointers into the argumentation issue base. The firing of a critic signals a breakdown to users and provides them with entry into the exact place in the argumentative hypertext system at which the corresponding argumentation is located.
- *ARGUMENTATION ILLUSTRATOR:* The explanation given in argumentation is often highly abstract and very conceptual. Concrete design examples that match the explanation help users to understand the concept. The ARGUMENTATION ILLUSTRATOR helps users to understand the information given in the argumentative hypertext by finding a catalog example that realizes the concept (Fischer, 1990).
- *CATALOG EXPLORER:* This helps users to search the catalog space according to the task at hand (Fischer & Nakakoji, 1991). It retrieves

design examples similar to the current construction situation and orders a set of examples by their appropriateness to the current specification.

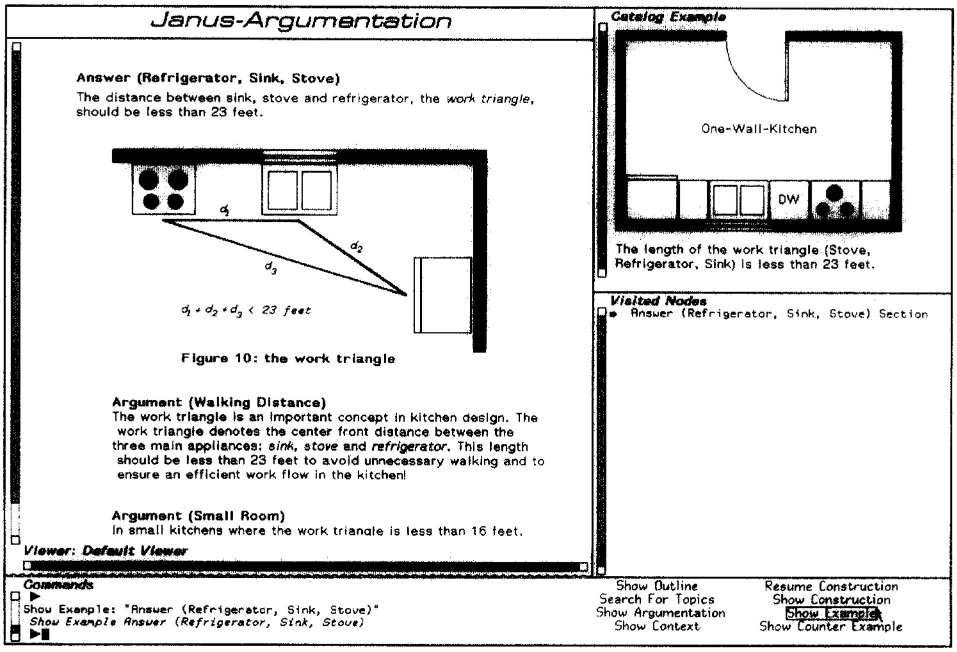
A typical cycle of events supported by the multifaceted architecture is: (a) users create and refine a partial specification or construction; (b) breakdowns occur; (c) users switch and consult other components in the system made relevant by the system to the partially articulated task at hand; and (d) users refine their understanding based on the back-talk of the situation. As users go back and forth among these components, the problem space is narrowed, a shared understanding between users and the system evolves, and the artifact is refined incrementally. This article focuses on the integration of construction and argumentation. Other components of the multifaceted architecture are described elsewhere (see Fischer, 1990; Fischer & Nakakoji, 1991).

***JANUS-ARGUMENTATION: The Argumentation Component of JANUS.*** JANUS-ARGUMENTATION is the argumentation component of JANUS (see Figure 5). It is an argumentative hypertext system based on the PHI method and implemented using the SYMBOLICS DOCUMENT EXAMINER (Walker, 1987). JANUS-ARGUMENTATION offers a domain-oriented, generic issue base about how to construct residential kitchens. This design knowledge has been acquired from protocol studies (Fischer et al., 1989b) and from kitchen design books (Jones & Kapple, 1984). In JANUS-ARGUMENTATION, designers explore issues, answers, and arguments by navigating through the issue base. The starting point for the navigation is the argumentative context triggered by a critic message in JANUS-CONSTRUCTION. Clicking with the mouse on a critique in JANUS-CONSTRUCTION (see Figure 3) activates JANUS-CONSTRUCTION and accesses the issue and answer corresponding to the critique. At any place in the issue base, designers can invoke the ARGUMENTATION ILLUSTRATOR to obtain an example from the catalog that implements the current issue answer.

## 5.3. Evaluation, Shortcomings, and Limitations of JANUS

***Evaluation.*** We have informally evaluated JANUS with subjects ranging from neophyte to expert designers and from neophyte to expert computer users (Fischer et al., 1989b). The subjects were tested in an experiment consisting of two tasks: a learning task and a design task. The learning task consisted of improving a "bad" kitchen design from the catalog (see Figure 3), and the design task consisted of designing a "good" kitchen, given a set of constraints. The constraints were imposed to test the various operations of the system. Users unfamiliar with the computer system were given help by the experimenter during an initial learning task. A final questionnaire was given to the subjects after the experiment.

***Figure 5.*** **JANUS-ARGUMENTATION: Rationale for the work triangle rule. JANUS-ARGUMENTATION is an argumentative hypertext system based on the PHI method. The Viewer pane shows a diagram illustrating the work triangle concept and arguments for and against the work triangle answer. The top right pane shows an example illustrating this answer generated by the ARGUMENTATION ILLUSTRATOR. The Visited Nodes pane lists in sequential order the previously visited argumentation topics. By clicking with the mouse on one of these items, or on any bold or italicized item in the argumentation text itself, the user can navigate to related issues, answers, and arguments. Hypertext access and navigation are made possible using this feature, inherited from the SYMBOLICS DOCUMENT EXAMINER.**


Janus-Argumentation

**Answer (Refrigerator, Sink, Stove)**
The distance between sink, stove and refrigerator, the *work triangle*, should be less than 23 feet.

**Figure 10: the work triangle**

**Argument (Walking Distance)**
The work triangle is an important concept in kitchen design. The work triangle denotes the center front distance between the three main appliances: *sink, stove* and *refrigerator*. This length should be less than 23 feet to avoid unnecessary walking and to ensure an efficient work flow in the kitchen!

**Argument (Small Room)**
In small kitchens where the work triangle is less than 16 feet.

***Viewer: Default Viewer***

***Catalog Example***

The length of the work triangle (Stove, Refrigerator, Sink) is less than 23 feet.

***Visited Nodes***
Answer (Refrigerator, Sink, Stove) Section

***Commands***
```
▶
Show Example: "Answer (Refrigerator, Sink, Stove)"
 Show Example Answer (Refrigerator, Sink, Stove)
▶▮
```

Designers with limited domain knowledge were able to understand the critics and learn from them to create reasonable kitchen designs. For example, several students did not know that building codes require at least one of the entrances into a kitchen to be at least 36 in. wide. One user also learned that the stove should be away from a door, based on safety requirements with respect to fire and burn hazard. He found this to be especially relevant to his own home where small children are constantly running in and out of the kitchen.

The critics were appreciated but were often ignored when they actively critiqued the user during construction. One user replied to this by saying that too much information was presented and that she could give attention to only one thing at a time. She preferred to complete some part of the design and then ask the system for a critique by using the **Critique All** command. Other users explained that they ignored critics because they already had been aware of them, either by a previous critique or by the fact that they already knew about them (such as that the sink should be in front of a window). In the questionnaire, all users found that critiquing was helpful in reminding them about design rules they did not think about while they were designing.

Users uncertain about a critique from the system or interested in more background information about design principles entered the hypertext system by clicking on the critique message. No users got lost in the hyperdocument, but one found that some of the arguments were not justified from his point of view. He would have liked to add his own counterarguments to it. Currently end-user modifications of the issue base are not supported. Another user found that some arguments did not go into enough depth in order to be persuasive. For example, he would have liked to know why a building code requires that a kitchen entrance should be greater than 36 in. wide.

***Shortcomings and Limitations.*** Our integrated design environments in their current form still suffer from a number of major limitations:

- Design environments need to evolve for the following reasons: (a) The world modeled in these design environments changes (Curtis, Krasner, & Iscoe, 1988), and (b) the background knowledge for a design domain cannot be articulated fully—It is tacit and requires breakdown situations to be activated (Ehn, 1988; Winograd & Flores, 1986). End-user modifiability is a prerequisite for evolution because the breakdown situations are experienced by the domain experts using these systems, not by the knowledge engineers who built them originally. Fischer and Girgensohn (1990) described a mechanism to make JANUS-CONSTRUCTION end-user modifiable. REFLACT (described in the next section) is an effort to make the argumentative component adaptive. One reason that JANUS-ARGUMENTATION failed to achieve this goal is that the DOCUMENT EXAMINER (Walker, 1987) is only a reader's interface to the hypertext system and requires a different writer's interface (Walker, 1988). Therefore, JANUS-ARGUMENTATION primarily serves as a design information system and does not allow the addition of new design rationale in a contextualized manner.
- The back-talk of the situation must be enhanced further with a simulation component providing us with insights that argumentation does not capture. This requirement became obvious in our experiments with professional kitchen designers who tested their designs by running mental simulations of specific situations (e.g., preparing a fancy dinner, imagining work-flow patterns with more than one person in the kitchen).

- The issue base of JANUS-ARGUMENTATION is generic; that is, it is used for any kitchen design project. The issue base is also static in that it does not adapt to the individual design projects. Some issues are only relevant to some of the design projects addressed by the issue base. For example, if the kitchen has no eating area, then issues relating to the eating area in the kitchen are irrelevant. Structures in the issue base irrelevant to the task at hand make the issue base unwieldy and make it difficult to find the relevant information. To filter out irrelevant information from a generic issue base, the serves relationship must be computed dynamically from the task at hand. The exploratory nature of design makes any static argumentative hypertext system, such as JANUS-ARGUMENTATION, inadequate. A dynamic hypertext system adapts to design decisions such as adding or removing an eating area.

## 6. CURRENT AND FUTURE WORK

Our current and future work is focused on four ways to make argumentation better serve design: (a) Static issue bases are being made extensible and dynamic, (b) the reusability of issue bases is being improved, (c) design rationale is being added to the examples in the catalog, and (d) a system component for articulating and representing the task at hand is being developed. Some of these extensions are being carried out as separate efforts later to be integrated into the overall environment.

***Adaptive Issue Bases.*** In response to the problems caused by the static nature of JANUS-ARGUMENTATION, we are exploring ways to make reusable issue bases more active and responsive to the situation, thus increasing the immediate benefit of issue bases. We have implemented these methods in REFLACT, a PHI-based hypertext system (Lemke, 1990). In REFLACT, the designer not only consults the issue base but also indicates design decisions—whether deliberated or not—by selecting one or more answers. The selected answers determine which issues the system raises from its issue base. This is done with the help of the PHI subissue relationship. In REFLACT, issue bases are fully modifiable and extensible by end users. Designers can add, modify, or delete issues, answers, and arguments without leaving REFLACT.

***Reusable Domain-Oriented Issue Bases.*** A design rationale is a large additional product of the design process. Creating and representing a design rationale is a great effort. Reuse of existing issue bases has the potential to reduce dramatically this effort. Every project is unique in some respects; few if any projects are unique in all respects. Therefore, the contents of a project issue base is not entirely unique to that project. Similar projects overlap substantially in issues, answers, and arguments. This is not to say that the issues are resolved in the same way, but merely that a great deal of the reasoning is shared by projects.

Reusable issue bases can serve as seeds that grow with each new design project. Each project extends and enhances the reusable issue base. The issue base being reused provides information about how to decompose the task, possible answers to issues, and principles of design. The issue base also warns designers of potential dead ends and unproductive solution directions. This is important because designers need better access to domain-oriented information (Curtis et al., 1988). Even expert designers can no longer master all the relevant knowledge, especially in technologically oriented design, where growth and change of the knowledge base are incessant (Draper, 1984; Norman, 1988).

Domain-oriented issue bases also amplify the designer's ability to reflect on issues. Recurring design issues could be researched intensively, and the results of this could then be stored at the appropriate location in the issue base for use by future designers encountering similar decisions in the future. This would, for example, allow the "folk theories" of designers to be subjected to rigorous scientific scrutiny. Cumulative domain-oriented issue bases could also foster communication among designers, researchers, and users about recurring matters of design.

The PHI subissue relationship is crucial to making issue bases reusable. The hierarchical grouping of issues allows argumentation systems to be built that filter issue bases according to the specifics of the new task. REFLACT filters issue bases using its mechanism of issue conditions. The system provides a common issue base for all projects in a domain such as kitchen design. This issue base includes issues, answers, and arguments at all levels of generality. As pointed out before, not every issue applies in each design project, even if it falls into one general domain.

***Enriched Catalogs.*** The JANUS catalog currently does not contain the design rationale for the designs it contains. By adding the rationale to each catalog example, designers can better understand the examples, can more easily find examples that are similar to the kitchens they are designing, and can reuse the rationale.

***Representation of the Task at Hand.*** More support to capture incrementally the task at hand is needed. Beyond the information contained in the construction situation, our specification component needs to be developed further to let designers articulate the specifics of their design efforts. This knowledge can be used by REFLACT to filter out irrelevant information from a reusable issue base. An initial effort in this direction is described by Fischer and Nakakoji (1991).

## 7. CONCLUSIONS

The title of this article, "Making Argumentation Serve Design," indicates that it is a challenge to make the use of design rationale feasible. We have identified major obstacles for meeting this challenge. Creating and using design rationale is a time-consuming process that must be carried out in addition to standard design activities, and there is little immediate reward. Recording and accessing design rationale can disrupt design and interfere with reflection in action. Argumentation that is removed from construction loses relevance to the task at hand. Without tight integration of argumentation and construction, designers fail to apply argumentation in the construction activity.

We have analyzed these problems within the design theories of Schön (1983) and Rittel (1984). These analyses gave us a constructive understanding that suggested the following solution approaches. First, the IBIS method had to be modified to emphasize relevance to the task at hand. This resulted in the development of the PHI method. Second, the PHI hypertext systems reduce the amount of secretarial work involved in managing issue bases. Third, support for reuse of issue bases reduces the conceptual work in creating project rationale and creates an issue base whose visible content and form correspond to the designer's changing understanding of the problem. Fourth, we developed tools for construction that support human problem-domain communication and integrated them with tools for argumentation via critics. These integrated design environments form a synergistic whole by causing the construction situation to talk back to the designer.

A final word on the generality of our approach is needed. JANUS was used as an "object to think with" in this article. We have used the same basic approach for user interface design (Lemke & Fischer, 1990), development and maintenance of Cobol programs (Atwood et al., 1991), river basin planning and operations (Lemke & Gance, 1990), computer network design (Fischer et al., 1991), knowledge editing, and design and planning of lunar habitation. As these systems get used in realistic work environments, we will get valuable feedback about the viability, the strengths, and the weaknesses of this approach.

***Acknowledgments.*** The authors thank the members of the Human-Computer Communication group at the University of Colorado who contributed to the conceptual framework and the systems discussed in this article.

***Support.*** The research was supported by the National Science Foundation under Grant Nos. CDA-8420944, IRI-8722792, and IRI-9015441; by the Army Research Institute under Grant No. MDA903-86-C0143; and by grants from the Intelligent Interfaces Group at NYNEX and from Software Research Associates (SRA) in Tokyo.

## REFERENCES

Akin, O. (1978). How do architects design? In J. Latombe (Ed.), *Artificial intelligence and pattern recognition in computer aided design* (pp. 65–119). New York: North-Holland.

Atwood, M. E., Burns, B., Gray, W. D., Morch, A. I., Radlinski, E. R., & Turner, A. (1991). The GRACE integrated learning environment—A progress report. *Proceedings of the Fourth International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems* (IEA/AIE 91; pp. 741–745). Tennessee: University of Tennessee Space Institute (UTSI).

Conklin, J. (1987). Hypertext: An introduction and survey. *IEEE Computer, 20*(9), 17–41.

Conklin, J., & Begeman, M. (1988). gIBIS: A hypertext tool for exploratory policy discussion. *Proceedings of the Conference on Computer Supported Cooperative Work,* 140–152. New York: ACM.

Cross, N. (1984). *Developments in design methodology.* New York: Wiley.

Curtis, B., Krasner, H., & Iscoe, N. (1988). A field study of the software design process for large systems. *Communications of the ACM, 31,* 1268–1287.

Draper, S. W. (1984). The nature of expertise in UNIX. *Proceedings of INTERACT '84, IFIP Conference on Human-Computer Interaction,* 182–186. Amsterdam: Elsevier Science Publishers.

Ehn, P. (1988). *Work-oriented design of computer artifacts.* Almquist & Wiksell International.

Fischer, G. (1990). Cooperative knowledge-based design environments for the design, use, and maintenance of software. *Proceedings of the Software Symposium '90,* 2–22. Kyoto, Japan.

Fischer, G., & Girgensohn, A. (1990). End-user modifiability in design environments. *Proceedings of the CHI '90 Conference on Human Factors in Computing Systems,* 183–191. New York: ACM.

Fischer, G., Grudin, J., Lemke, A. C., McCall, R., Ostwald, J., & Shipman, F. (1991). *Supporting collaborative design with integrated knowledge-based design environments* (Technical Report). Boulder: University of Colorado, Department of Computer Science.

Fischer, G., & Lemke, A. C. (1988). Construction kits and design environments: Steps toward human problem-domain communication. *Human-Computer Interaction, 3,* 179–222.

Fischer, G., McCall, R., & Morch, A. (1989a). Design environments for constructive and argumentative design. *Proceedings of the CHI '89 Conference on Human Factors in Computing Systems,* 269–275. New York: ACM.

Fischer, G., McCall, R., & Morch, A. (1989b). JANUS: Integrating hypertext with a knowledge-based design environment. *Proceedings of Hypertext '89,* 105–117. New York: ACM.

Fischer, G., & Nakakoji, K. (1991). Empowering designers with integrated design environments. *Proceedings of the First International Conference on Artificial Intelligence in Design,* 191–209. Edinburgh, UK: Royal Museum of Scotland.

Hutchins, E. L., Hollan, J. D., & Norman, D. A. (1986). Direct manipulation interfaces. In D. A. Norman & S. W. Draper (Eds.), *User centered system design, new perspectives on human-computer interaction* (pp. 87–124). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Jones, R. J., & Kapple, W. H. (1984). *Kitchen planning principles–equipment–appliances.* Urbana-Champaign: University of Illinois, Small Homes Council–Building Research Council.

Kunz, W., & Rittel, H. (1970). *Issues as elements of information systems* (Working Paper No. 131). Berkeley: University of California, Center for Planning and Development Research.

Lemke, A. C. (1990). Framer-hypertext: An active issue-based hypertext system. *Proceedings of the Workshop on Intelligent Access to Information Systems,* 34–38. Darmstadt, Germany: GMD-IPSI.

Lemke, A. C., & Fischer, G. (1990). A cooperative problem solving system for user interface design. *Proceedings of AAAI-90, Eighth National Conference on Artificial Intelligence,* 479–484. Cambridge, MA: AAAI Press/MIT Press.

Lemke, A. C., & Gance, S. (1990). *End-user modifiability in a water management application* (Tech. Rep. No. CU-CS-541-9). Boulder: University of Colorado, Department of Computer Science.

McCall, R. (1979). *On the structure and use of issue systems in design* (Doctoral Dissertation, University of California, Berkeley, 1978). University Microfilms.

McCall, R. (1991). PHI: A conceptual foundation for design hypermedia. *Design Studies, 12,* 30–41.

McCall, R., Bennett, P., d'Oronzio, P., Ostwald, J., Shipman, F., & Wallace, N. (1990). PHIDIAS: A PHI-based design environment integrating CAD graphics into dynamic hypertext. In A. Rizk, N. Streitz, & J. André (Eds.), *Hypertext: Concepts, systems and applications: Proceedings of the European Conference on Hypertext, INRIA* (pp. 152–165). Cambridge, England: Cambridge University Press.

McCall, R., Mistrik, I., & Schuler, W. (1981). An integrated information and communication system for problem solving. In H. S. Glaeser (Ed.), *Data for science and technology—Proceedings of the Seventh International CODATA Conference 1980* (pp. 512–516). London: Pergamon.

Neches, R., Swartout, W. R., & Moore, J. D. (1985). Enhanced maintenance and explanation of expert systems through explicit models of their development. *IEEE Transactions on Software Engineering, SE-11,* 1337–1351.

Norman, D. A. (1986). Cognitive engineering. In D. A. Norman & S. W. Draper (Eds.), *User centered system design, new perspectives on human–computer interaction* (pp. 31–62). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Norman, D. A. (1988). *The psychology of everyday things.* New York: Basic Books.

Paradies, K. (1973). *The kitchen book.* New York: Wyden.

Reuter, W., & Werner, H. (1983). *Thesen und Empfehlungen zur Anwendung von Argumentativen Informationssystemen* [Theses and recommendations about the use of argumentative information systems] (Working Paper). Stuttgart: University of Stuttgart, Institute for Foundations of Planning. (To be published in English in D. Noble (Ed.), *Issue-based information systems,* Prentice-Hall)

Rittel, H. W. J. (1972). On the planning crisis: Systems analysis of the first and second generations. *Bedriftsokonomen, 8,* 390–396.

Rittel, H. W. J. (1984). Second-generation design methods. In N. Cross (Ed.), *Developments in design methodology* (pp. 317–327). New York: Wiley.

Schön, D. A. (1983). *The reflective practitioner: How professionals think in action.* New York: Basic Books.

Simon, H. A. (1981). *The sciences of the artificial.* Cambridge, MA: MIT Press.

Stefik, M. J. (1986, Spring). The next knowledge medium. *AI Magazine,* pp. 34-46.

Walker, J. H. (1987). Document examiner: Delivery interface for hypertext documents. *Hypertext '87 Papers,* 307-323. Chapel Hill: University of North Carolina.

Walker, J. H. (1988). Supporting document development with Concordia. *IEEE Computer, 21*(1), 48-59.

Winograd, T., & Flores, F. (1986). *Understanding computers and cognition: A new foundation for design.* Norwood, NJ: Ablex.

***HCI Editorial Record.*** First manuscript received July 3, 1990. Revisions received February 14, 1991, and May 6, 1991. Accepted by John M. Carroll. —*Editor*