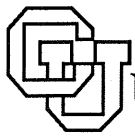
A Computational Medium for Supporting Interpretation in Design

Gerry Stahl

CU-CS-598-92 Revised August 1992



University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

A COMPUTATIONAL MEDIUM FOR SUPPORTING INTERPRETATION IN DESIGN

Gerry Stahl

CU-CS-598-92

June 1992

rev. 8-5-92

Submitted to:	*Journal of Architecture and Planning Research*
For the special issue on:	"Computational Representations of Knowledge"
Edited by:	Mark Gross
For publication in:	June, 1993

Department of Computer Science
University of Colorado at Boulder
Campus Box 430
Boulder, Colorado 80309-0430 USA

phone: (303) 444-2792
email: gerry@cs.Colorado.edu

A COMPUTATIONAL MEDIUM FOR SUPPORTING INTERPRETATION IN DESIGN

by Gerry Stahl

Abstract

Theorists of design methodology have described facets of design and problem solving that call for computer support. However, their assessments conflict in fundamental ways with the techniques of artificial intelligence. For instance, pivotal writings about design by Alexander, Rittel and Schoen argue that representations of a problem must grow out of the designers' engaged intuitions, through deliberations among opposing views, and as a result of interactions with the concrete design situations. Artificial intelligence, however, traditionally plays upon the computational power of *a priori* formal representations. This creates a troubled tension between human *interpretation* and computer *representation*.

Heidegger's hermeneutic philosophy of interpretation provides a framework for analyzing aspects of the design process that have been systematically overlooked by rationalist approaches to computerization. It suggests ways in which the interpretive processes at the heart of creative designing can be supported by software. However, the construction of representation systems must be a part of the design process, under human control.

Hermes is a computer system developed to support design of lunar habitats. This research prototype features a special language for defining terms, conditions, critics, and queries to display design information from the interpretive perspectives of human users. Hierarchies of interpretive contexts facilitate the sharing of these perspectives. All textual, graphical, and other information is integrated and inter-related by a form of hypermedia incorporating these language and context mechanisms. This provides a computationally active medium for expressing, storing, communicating and critiquing design interpretations and representations.

The message of Hermes is that computers can support human creativity in design rather than automating or rigidifying the design process. To do this, a new approach to software is needed that heeds the deeper principles of design methodology and the nature of human interpretation.

A COMPUTATIONAL MEDIUM FOR SUPPORTING INTERPRETATION IN DESIGN

Table of Contents

A COMPUTATIONAL MEDIUM FOR SUPPORTING INTERPRETATION IN DESIGN

Introduction: the Problem with Computational Representations of Knowledge

Implicit in the special issue theme, Computational Representations of Knowledge, lies a question: How can knowledge of activities like architecture and planning be represented in computer systems so that these systems can compute solutions to design problems? Attempts within AI (artificial intelligence, the discipline of computer science concerned with such matters) to attack this question straight on have failed. Expert systems, for instance, have tried to represent the knowledge of domain experts in systems of propositional rules and then to compute solutions via logical inference, but the results have been disappointing. (Winograd and Flores, 1986)

Perhaps previous attempts to computerize design failed because they operated within the framework of a rationalist philosophy which is inappropriate for design. That philosophy makes assumptions about computation, representation and knowledge which have proven very successful in the development of the natural sciences. However, when creating computer applications for domains of design it may be necessary for AI to develop a new paradigm based more on a philosophy of the (*verstehende* or "hermeneutic") human sciences. (Suchman, 1987) Such an approach might recognize that designers do not explicitly represent most of their knowledge in a propositional form (Polanyi, 1962), that personal interpretations play a more central role than objective knowledge (Dreyfus, 1972), and that the computer may be better suited to augmenting the deliberations of human decision-making than to computing solutions autonomously (Engelbart, 1963).

This paper explores the possibility and desirability of such an alternative by way of an interpretive reflection upon the traditions in which it is situated: design methodology and artificial intelligence. Within design, it tries (in Section I) to uncover the thread of an argument that defines the limits of technical rationality for this field. In doing so, it focuses on insights of Alexander, Rittel and Schoen. These three writers have all been concerned with the problem of how to use computers in design work, although they may at times have been disillusioned by the difficulties to such an extent that they sometimes felt computers were counterproductive for designing (e.g., Alexander, 1971). Perhaps computer technology was simply not adequate to the task in the past, or perhaps it was necessary for the rationalist approach to be carried forth and to fail before another option could be seriously explored.

At any rate, there are now signs within the AI field itself of a movement in the new direction. These are followed (in Section II) through a shift from expert systems to computer critiquing systems, and from these to design environments. The need for an alternative to the traditional approach becomes particularly pronounced as certain kinds of domains are increasingly being addressed. The expert system paradigm assumes that domain knowledge can be codified, and expert systems have been most successful in fields which are well understood in terms of established theory. However, for "exploratory" tasks which are now seeking computer support -- like lunar habitat design -- designing is more a matter of making up the rules and definitions than following existing formulae; here the established AI methods are clearly inadequate.

The research underlying this paper was initiated in response to the concerns of a design firm that contracts with NASA for projects related to missions to the moon. They sought help in developing computer tools for managing the burdensome volume of information that had to be kept in mind during their design work. The first stage in the participatory design project of developing useful software was to understand their current practices. (Ehn, 1988) This was accomplished by videotaping a series of sessions in which a lunar habitat was designed. Analysis of these tapes showed that what was taking place was not simply the satisfaction of multiple constraints, which could be modelled by objective rules, but above all the *interpretation* of criteria, priorities and terminology. This empirical finding coincided well with theoretical considerations from the critique of AI, and established the direction of research.

The title of this paper suggests an alternative to the traditional implications of Computational Representations of Knowledge that emerged through the research. It replaces the emphasis on objective, propositional knowledge with one on designers' creative interpretations. It suggests that computer systems should provide a medium of external memory for designers -- an electronic extension of writing or sketching paper -- rather than trying to compute solutions. It supports designers, rather than replacing the human element. A model of this alternative paradigm of AI is presented (in Section III) in the form of Hermes, the prototype software system developed to support design of lunar habitats. Hermes features a disclosure language for designers to define ways of revealing relevant information and communicating design ideas. It also provides a system for organizing interpretive contexts, so that multiple personal perspectives on design artifacts can be elaborated and shared. The language and context mechanisms are integrated in an extended form of hypermedia, to form a computationally powerful medium of external memory for designers to create, explore, communicate and store their concepts and plans.

Hermes exemplifies a new departure for AI, one founded on an analysis of human interpretation rather than on the logic of formalizable propositional knowledge. It respects the irreducible power of the human mind to understand meaning and to explicate understanding. Therefore, it strives to provide computational media which can augment human interpretation, rather than providing Computational Representations of Knowledge. The message of Hermes is that computers can support human creativity in design, instead of automating or rigidifying the design process. This paper tries to show how the approach of Hermes heeds the deeper principles of design methodology and the nature of human interpretation in order to provide computational power to enhance design without restricting personal innovation.

I. Design Methodology from the Perspective of Computer Support

Alexander: balancing computation with human intuition

Deliberation on the question of whether and how computers should be used to support the work of designers has raged for several decades now. The issues go to the heart of what design is and should be. In his now classic *Notes on the Synthesis of Form*, Christopher Alexander reviewed the history and even the prehistory of design in order to argue that the field reached a second watershed in the mid-twentieth century. The profession of design had originally emerged when society started to produce new needs and innovative perspectives too rapidly to allow forms to be developed through "unselfconscious" activities of slowly evolving traditions. Now, the momentum of change has reached a second qualitatively new stage:

> Today more and more design problems are reaching insoluble levels of complexity. This is true not only of moon bases, factories, and radio receivers, whose complexity is internal, but even of villages and teakettles. In spite of their superficial simplicity, even these problems have a background of needs and activities which is becoming too complex to grasp intuitively. (Alexander, 1964, p.3)

The management of complexity must become a primary concern of the field of design. The level of complexity that Alexander had in mind is characterized by the fact that it exceeds the ability of the unaided individual human mind to handle it effectively. Various methodologies can help to decompose complexity, and this is where the mathematical structures, diagrams or patterns that Alexander proposed come in. They provide the representational or computational basis today for computerization.

Alexander saw a major advantage of the systematic use of these structures in what he referred to as a "loss of innocence". When design first became a profession with rules that could be stated in language and taught, there was, according to Alexander's account, a first such loss of innocence. More recently, when Bauhaus designers recognized that one could design for mechanized production, another accommodation was made with changing times. The use of systematic methodologies to help manage complexity would, Alexander claimed, entail an analogous acceptance of the limitations of the individual designer's intuitive powers. This would bring with it a significant opportunity for progress of the profession. When the design process is formulated in terms of abstract structures it becomes much more readily subject to public criticism than when it is concealed in the mysteries of the lonesome genius' artistry, just as the earlier formulation of previously unselfconscious design into explicit plans, articulated processes and stated justifications laid the basis for a science of design which could be refined through on-going debate. Loss of innocence entails the removal of an outmoded barrier to the kind of public critical reflection required for a profession.

But Alexander did not see the issue one-sidedly. Recognizing the power of both formal representations and non-formalizable tacit knowledge, he did not propose that design methods substitute for the practice of design or for the designer's practical intuitions. Rather, he recognized that intuition and rationalism were equally necessary, and argued for a proper balance: "Enormous resistance to the idea of systematic processes of design is coming from people who

recognize correctly the importance of intuition, but then make a fetish of it which excludes the possibility of asking reasonable questions." (p. 9) Alexander felt that the fetishism of intuition as some kind of inalienable artistic freedom of the designer functioned as a flimsy screen to hide the individual designer's incapacity to deal with the complexity of contemporary design problems. As a consequence of the designer ignoring these limitations, the unresolved issues of complexity get passed down to engineers who have been trained to work out details rather than to grasp complex organization synthetically; the product that results tends to be a monument to the personal idiom of the creator rather than an artifact with a good fit to its function.

The questions posed by Alexander three decades ago for design methodology generally still confront the particular task of figuring out how best to use computers to support designing. Consider his first example above, that of designing a moon base. Clearly, this is an overwhelmingly complex task. One needs to take into account technical information about supporting humans in outer space, including issues that may not have previously been thought of and investigated (such as the practicality of using lunar rocks as building materials). One must also consider the mission goals of the base, both stated and implicit. Then there are social and psychological issues concerning the interactions among groups of people who are confined in an alien environment for a prolonged period of time. All of these factors interact with the more common issues of designing a habitat for working, eating, socializing, and sleeping -- resulting in a design problem of considerable complexity. While computers may be necessary to manage this complexity, the tacit knowledge of human designers must also be brought to bear with their intuitions about what it would be like to live together in a lunar habitat.

Rittel: tackling wicked problems through argumentation

When Horst Rittel declared in his *Dilemmas in a General Theory of Planning* that "planning problems are inherently wicked," (Rittel & Webber, 1972, p.10) he thereby spelled out that characteristic of planning and design tasks that has subsequently become the central source of perplexity in trying to imagine a computer system that can effectively support the challenging aspects of design. For, computer programs have traditionally been devised in accordance with the classical example of "tame" science and engineering problems -- precisely the paradigm that Rittel argued is not applicable to the problems of open societal systems with which planners and designers are generally concerned. This assumes that a problem can first of all be formulated as an exhaustive set of specifications. Then, based on such a problem statement, possible solutions can be evaluated to see which are optimal solutions to the problem. Computer programs based on this paradigm must represent in advance the space of problems and solutions for a well-defined type of design problem in an explicit, comprehensive, and non-controversial (objective) manner.

Rittel claimed that the wicked problems of planning could not be thoroughly understood in the first place until one had already started to explore directions for solutions. Suppose, for instance, that you are asked to plan a mission to the moon for four astronauts for a period of 45 days. According to NASA, the purpose has been specified as: to explore long-term stays for crews of international backgrounds and mixed gender and to conduct some scientific research and some site work to prepare for future moon bases. In thinking about the design of the lunar habitat for this mission, you might begin to discuss the importance of privacy issues with other people on

your design team. You might feel that not only was some physical privacy needed for cultural reasons, but psychologically there would be a need to structure a careful mix of public and private spaces and opportunities. These privacy issues might become paramount to your design even though they had not been included in the original problem statement. In this way, the set of issues to be investigated and concerns to be balanced would emerge and evolve as the planning process took place.

In opposition to the then dominant methods of operations research which tried to compute optimal solutions from static and well-defined ("tame") problem statements, Rittel called for a model of planning as "an argumentative process in the course of which an image of the problem and of the solution emerges gradually among the participants, as a product of incessant judgment, subjected to critical argument." (p. 13) The language used in real, significant planning processes is itself the result of discussion and debate among various parties, each of whom uses subjective judgments to criticize hidden assumptions and to reconstrue implicit meanings of terms. No one view of the problem or its solution has a necessary priority. The framing of problems and the judging of solutions arise through critique, deliberation and reinterpretation, not by inference from an objective viewpoint. For Rittel, people's perspectives on problems are necessarily based on subjective conditions such as their individual value systems and political commitments or their personal roles *vis a vis* the proposed solutions:

> For wicked planning problems, there are no true or false answers. Normally, many parties are equally equipped, interested, and/or entitled to judge the solutions, although none has the power to set formal decision rules to determine correctness. Their judgments are likely to differ widely to accord with their group or personal interests, their special value-sets, and their ideological predilections. (p. 15)

Consider again the concept of privacy in the lunar habitat. A design team might start from the idea of visual privacy. Through discussion of the implications of life in this confined space, they might want to include protection from the noise of flushing toilets and snoring neighbors. But then the team member concerned with medical contingencies might introduce a notion of privacy for an injured astronaut who needs to recuperate. Psychologists, sociologists, engineers and other members of the design team would each come to the common task with different perspectives. Given a methodology which builds on the strengths of design as an argumentative group process, these differences can contribute to a robust solution that takes into account a variety of competing and interacting insights, not all of which could have been anticipated in advance.

Computer support for planning and design processes as Rittel conceived of them must allow team members to articulate their individual views and judgments, to communicate these to each other, and to forge shared perspectives. It must support deliberation or argumentation. Rittel himself made some initial attempts to define computerized issue-based information systems, leading to recent systems like gIBIS (Conklin, et al, 1988) and Mikroplis (McCall, 1989). Somehow, the dimensions of the design problem must be allowed to emerge and change as different perspectives are brought to bear, as initial approaches are subjected to critique, and as solutions gradually emerge. Computer systems may be useful for storing, organizing and communicating complex networks of argumentation -- as long as they do not stifle innovation by imposing fixed representations of the ideas they capture or limiting diversity of interpretive viewpoints.

Schoen: dialogues of discovery and creation

Alexander and Rittel have suggested the importance of the individual designer's intuitions and of public processes of deliberation for the development of good design. This is at least implicitly a rejection of the model of technical rationality based on the methodology of the natural sciences. Donald Schoen made this rejection even more explicit in his influential study of the design profession, *The Reflective Practitioner* (1983). Here he argued that much design knowledge is tacit, rather than being rule-based. He viewed the design process as a dialogue-like interaction between the designer and the design situation, in which the designer makes moves and then perceives the consequences of these design decisions in the design situation (e.g., in a sketch). The designer manages the complexity that would be overwhelming if all the constraints and possibilities were formulated as explicit symbolic rules by using professionally-trained skills of visual perception, graphical sketching and vicarious simulation. Note that these skills by-pass the process of analyzing everything into primitive elements and laying it out in words and propositions.

Schoen recently took up the question of computer support for design in a paper with the descriptive title, *Designing as Reflective Conversation with the Materials of a Design Situation.* In this article he argued for a necessarily limited role for computers in design because one of the most important things that designers do is to create the design situation itself. Not only is this something that computers cannot do by themselves, but it also precludes computer programmers from pre-defining a generic design situation for the computer, prior to the involvement of the designer with the task.

Before trying to discuss potential computer roles, Schoen takes time to review several experiments supporting his thesis that designers construct the design situation. In one experiment, several experienced architects are shown a 14-sided, dimensioned polygon with door locations indicated, and asked to design a library with that shape as its footprint. One architect saw the figure in terms of simple end entrances and complex middle entrances; another saw it as three pods surrounding a middle; a third saw two Ls back to back. Clara, another subject, discovered a five foot displacement in the layout which complicated the spatial relationships considerably for her. (See Figure 1.) Schoen concludes from these and other studies that designers construct the problem by seeing the situation *as* defined in a certain way:

> In one sense, the 5 ft displacement that Clara noticed is there to be discovered. However, not everyone who tried the library exercise discovered it. Clara did. She noticed it, named it, and made a *thing* that became critically important for her further designing. In this sense, her treatment of the library exercise shows her not only discovering but *constructing* the reality of a design situation. For designers share with all human beings an ability to construct, via perception, appreciation, language and active manipulation, the worlds in which they function. . . . Every procedure, and every problem formulation, depends on such an ontology: a construction of the totality of things and relationships that the designer takes as the reality of the world in which he or she designs. (Schoen, 1992, p.9)

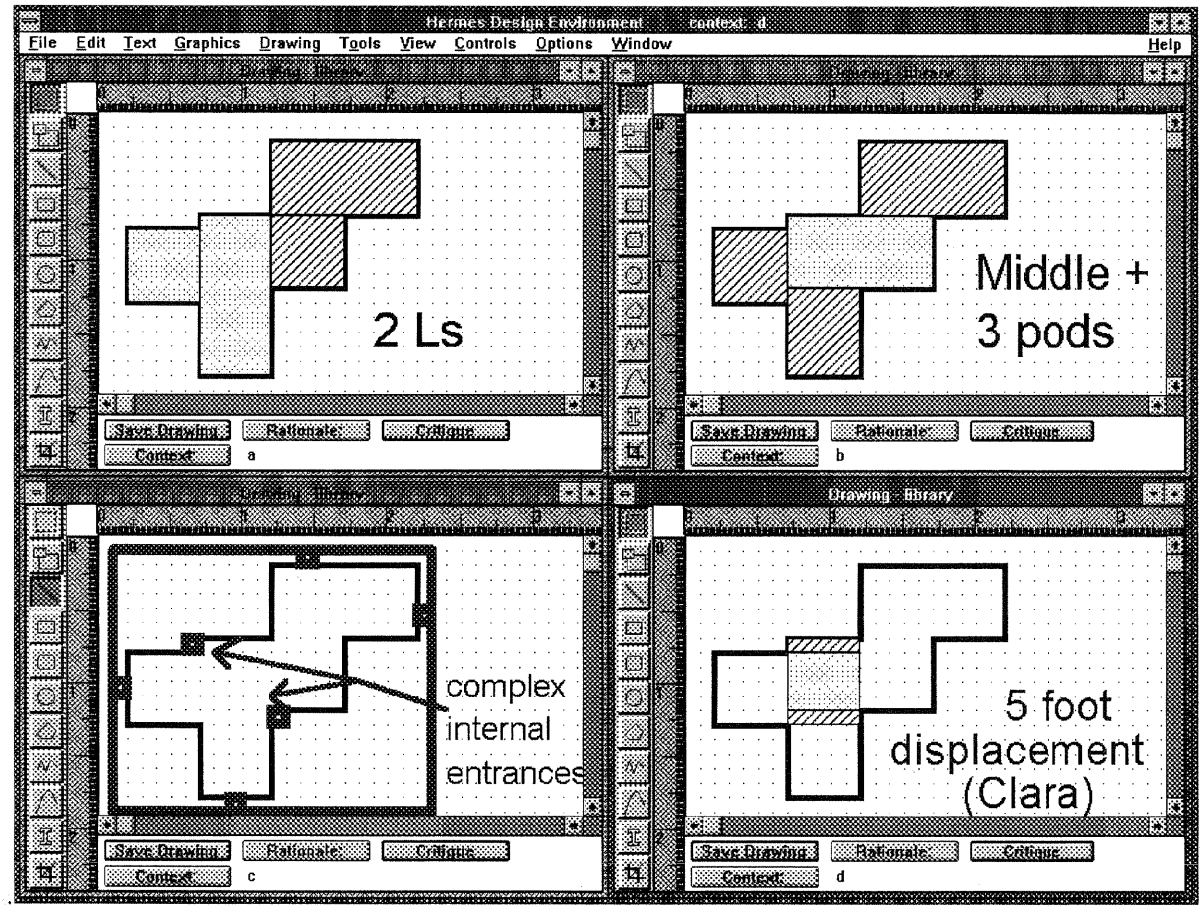


Figure 1. Four interpretations of the library.

Other experiments showed that designers also construct the materials, site, and relationships (or prototypes) in a similar way to how Clara constructed the crucial patterns of the project. In this sense, then, there is no given design problem which is explicitly and exhaustively defined before the designer comes to it. Correspondingly, there can be no well-defined problem space for the designer (or for some automated version of the designer) to search through methodically. Rather, the designer's subjective, personal or intuitive appreciations shape the problem by constructing its patterns, materials and relationships. The design project is solved by the designer experimenting with tentative moves within the constructed design situation and discovering the consequences of those moves.

Schoen argues that a computer program cannot on its own construct a design situation the way an architect does, picking out, naming, and focusing upon critical patterns, materials and relationships. To the extent that the role of a designer includes applying intuitive, perceptual and linguistic skills to view the situation creatively and to converse with it reflectively, a computer cannot do what a human designer does. Assuming that Schoen is correct that these skills are necessary for real design, a computer can also not accomplish the design task using alternative methods to those used by humans, because computer programs as we know them are ultimately

based on predefined representations of fixed and strictly delimited ontologies. Computer programs for design are therefore limited to solving problems in well-defined microworlds in which the framing of problems is trivial, or else to working with human designers to augment their tacit skills and to allow them to define the perspectives and concepts in terms of which tasks are to be undertaken.

The hermeneutics of design

Adrian Snodgrass and Richard Coyne of the Faculty of Architecture in Sydney have begun to articulate a new philosophical basis for AI in design by suggesting that design is hermeneutical. (For a similar move from the AI community, see Winograd & Flores, 1986) "Hermeneutics" is the study of interpretation, and today refers primarily to the philosophy of Martin Heidegger and its explication by Hans-Georg Gadamer. Snodgrass and Coyne argue that design is a human science in contrast to a natural science, and therefore must be founded on human understanding rather than on objective method. This has profound implications for the attempt to provide computer support for design, as well as for the more general attempt to comprehend the design process.

The ideas of hermeneutic philosophy provide a conceptual framework for further explicating Alexander's ideas about intuition and public critique, Rittel's views on wicked problems and the need for deliberative processes involving personal interests, and Schoen's analysis of tacit knowledge and the designer's dialogue with the constructed design situation. As a human science, design is based in human understanding gained through processes of interpretation (Alexander's intuition), rather than being based in knowledge, that is, in propositions and explicit rules. In fields like design, claims are not proven by appeal to objective facts and rigorous methods, but by reference to further interpretations (Rittel's argumentative process). A given claim reflects a certain interpretation of the design situation, a certain way of seeing it or constructing it (as Schoen would say).

It is always legitimate to question a design move and to demand some justification. But the justification will always be from the perspective of an interpretation, which can be questioned further. There are no axiomatic starting or stopping positions, such as those sought by the rationalist tradition. No claims form *a priori* foundations for arguments which cannot themselves be questioned; the chain of justifications based on interpretations ends only when one concedes that the argument is plausible or convincing from the perspective that one has been persuaded to adopt. This means the AI dream of providing well-defined representations for problem statements, rigorous algorithms for problem solving, and expert criteria for solution evaluation is misguided. Instead, ways should be found for computers to help people devise, share and debate innovative interpretations. But AI theory is based on formal logical reasoning (Nilsson, 1980) and provides little insight into subjective interpretation.

In philosophical terms, the problem with most AI systems is that they assume that a single interpretive framework can, at least in theory, be formulated that will be adequate for all representations within a given domain. This position can be traced back to Kant. In his *Critique of Pure Reason* (1787), Kant argued that the human mind imposes a set of elements or categories

on sense data in order to understand the external world. These elements or categories of space, time, quantity, quality, etc. that Kant derived were claimed to be universal *a priori*.

However, the universality of our interpretive framework was soon criticized by Hegel, who argued that reason evolved through history. In the *Phenomenology of Mind* (Hegel, 1807), for instance, Hegel laid out the logical stages of reason's development in terms of a review of human history. Marx, in turn, tied this idealist history to the social development of production relations in *Capital* (Marx, 1867), while Freud related it to the individual's formative history of interpersonal relationships (Freud, 1952).

Finally, Heidegger generalized these historical perspectives by saying that we always understand from within the situation in which we find ourselves already thrown as a result of our past. But he also added a second important dimension to this critique of Kant. Our interpretive perspective, he argued, is not simply a matter of categories that can be made explicit and stated in propositions. More fundamentally, it is a matter of understanding what it means to be a person and what it means for other things to be encountered in the world (Heidegger, 1927; see also Dreyfus, 1985). It is only in terms of this ontological preunderstanding -- which can be seen in the intentionality of our actions, in our grasp of linguistic meaning, in bodily adeptness, and in our interpersonal skills -- that we can in the first place make things explicit and formulate propositional knowledge.

So Heidegger's hermeneutic critique of Kantian rationalism has two central implications for the design of software to augment human cognition. First, it must not impose a conceptual framework or representation of domain ontology as though a universal set of categories could be formulated prior to experience with the particular problems in the domain. Second, computers can never take on the entire interpretive task because they can only manipulate explicit symbols, whereas interpretation requires a preunderstanding consisting of tacit background understandings which cannot be represented in formal rules or propositions because they form the very precondition for propositional knowledge.

Heidegger's thought offers a sustained critique of the rationalist tradition underlying AI, and proposes a philosophy oriented on the nature of human interpretation. (Stahl, 1975) According to Heidegger, interpretation takes place on the basis of three dimensions of preliminary understanding:

* Pre-owning: we already have a wealth of tacit, culturally acquired skills and practices that we bring with us as historical beings, and which makes us who we are.
* Fore-sight: we see our situation in terms of a conceptual framework and language in terms of which things can be disclosed to us.
* Pre-conception: we have a tentative expectation or anticipation of what it is that we are about to interpret.

Heidegger's analysis of preunderstanding incorporates within a consistent philosophic conceptualization many of the ideas important to Alexander, Rittel and Schoen: tacit (non-propositional) skills, linguistic problem framing, and discovery through construction.

Most of the time we form interpretations without being aware of this three-fold background of preunderstanding. That is why the interpretive process of design often seems so mysterious and

intuitive. As we are forced to justify or reflect critically upon the assumptions of our interpretive stance, we gradually make more and more of the underlying background explicit. We can be prompted to do this by what Schoen calls "breakdowns" in the design process. For instance, we make a move in our design sketch and then we see that a problem occurs as a result. Perhaps seeing the problem brings to our attention a certain need or constraint of the project that has been violated and that we were not formerly aware of. Breakdowns of relationships in our situation are a common way in which our circumstances are explicitly disclosed to us. Dialogue with other team members is another way in which implicit assumptions are brought to light, in explaining and arguing for ones own views in order to bridge the gap to someone else's perspective. Critical self-reflection while engaged in a design task is yet another way:

> The process of design is thus a disclosure, in two senses. Firstly, it is a disclosing of the artifact that is being designed; and secondly, and simultaneously, it is an unfolding of self-understanding, since it reveals one's preunderstandings. It uncovers the preconceptions that are constitutive of the design outcome, and at the same time brings to light the prejudices that are constitutive of what we are. (Snodgrass & Coyne, 1990, p. 15)

This conception of design as a dialogue which discloses involves a very different notion of critique from that in philosophy of science. The model is one of persuasion, not of hypothesis testing. One is always already in an interpretive context. From within this context, one then understands new arguments, claims, and interpretations. Being in an interpretive context is not like tentatively accepting a propositional conjecture that one may later flatly refute as false based on some discovered objective facts (Popper, 1965). It is more like having a perspective through which one can first of all understand arguments and facts, and thereby modify ones own framework or paradigm (Kuhn, 1962).

The notion of language operative here is also in contrast to that of the natural sciences and AI: "On the one hand there is the model of formalized language, the language of primary units that are combined according to the rules of logic to form meaningful structures; and on the other hand there is the metaphor of the language of conversation, which is the language of interpretation." (Snodgrass & Coyne, 1990, p. 16) This presents a serious problem for any attempt to provide computer support for design. Computers speak the formalized language, while designing requires the language of conversation. Computer programs consist quite precisely of algorithms encoded in a formal language, data structured as primary units, and operations performed in accordance with the rules of logic.

Snodgrass and Coyne have shown that design can be comprehended as a hermeneutic or interpretive endeavor. This view is consistent with the tradition of design methodology presented in the preceding pages. Like Alexander, Rittel and Schoen, Snodgrass and Coyne see their analysis as being relevant to AI, but when they try to spell out practical recommendation they end with vague generalities (See Coyne & Snodgrass, 1991). Hermes, on the other hand, is a working model of hermeneutic software design. However, before discussing how Hermes embodies the principles of hermeneutics in its specific features and implementation, it will be useful to indicate a trend within recent developments of AI that can be taken as converging with the suggestions from design methodology and preparing the path taken by Hermes.

II. Adapting Artificial Intelligence to Design

Simon: defending the traditional approach

Most work in the mainstream of AI can be characterized as an attempt to create computer programs to solve problems by using formalized language, primary units and the rules of logic. (Winston, 1981) In an article on *The Structure of Ill-structured Problems,* Herbert Simon, a major proponent of this tradition, tried to defend its approach against the argument presented above that the AI style of representing problems is inadequate given the "wicked" nature of most interesting design tasks. Simon's counter-argument revolves around the example of a chess playing program. He claims that the problem for such a program shifts from move to move as the features of the board (attacks, opportunities, strengths) change. So even chess is a wicked problem. Yet, a computer can play chess using traditional AI techniques. Therefore Simon claims there is no reason to think that wicked problems cannot be solved by these techniques.

But Simon has confounded two layers of representational analysis. Chess is a well-defined domain with explicit, unambiguous rules. In no sense does a chess program reinterpret the rules as the game proceeds. The representation of game states and therefore the universe of possible chess moves is fixed for all games. The fact that characteristics of the board's state and its evaluation change has no consequences for how one represents states or rules in chess. This is essentially different from how ones perspective on moves in social planning and complex design tasks has fundamental implications for how one frames problems and judges solutions.

At the end of his article, Simon provides a glimpse of the real issue. If a computer program needs to acquire external information about the problem situation, then it must force that information into its fixed representational framework. Simon admits that this is a weakness, but concludes that it is really for the best:

> [The process of acquiring external information] is an aid [to the process of understanding that information] because it fits the new information to formats and structures that are already available, and adapts it to processes for manipulating those structures. It is a limitation, because it tends to mold all new information to the paradigms that are already available. The problem-solver never perceives the *Ding an sich*, but only the external stimulus filtered through its own preconceptions. . . . The world perceived is better structured than the raw world outside. (Simon, 1973, p. 163)

The reference here to the Kantian "thing-in-itself" signals Simon's outmoded philosophic position, which ignores the need for representation to be mediated by what is represented. The whole point of Rittel's analysis of wicked problems was that there is no adequate set of formats and structures already available before one acquires information about a situation. Rather, an argumentative process is needed to respond to the flow of information in ways which transform the paradigms that were already available. Schoen's reflective conversation with the materials of a design situation makes no sense if the materials have been fit to a mute format. Although Heidegger would agree that the world is perceived through existing preconceptions, he would not agree that this is a "better" structure if the tentative original expectations are not allowed to respond and be transformed by the world they disclose.

Perhaps Simon realized that planners and designers need to take approaches that are qualitatively different from the methods of traditional AI, but he could not imagine how to extend computer technology to support those activities. In a lecture on *Social Planning*, he later recited a series of anecdotes that illustrated how complex planning processes hinge in large part on not assuming a fixed representation of the problem, but letting it evolve with the solution. For instance, in establishing the Marshall Plan after World War II the people involved in setting it up proposed six different and largely contradictory conceptions for its role. Simon underscores the observation that different conceptualizations of the problem would imply various ways of organizing the agency, and consequently quite different programs emphasizing different results. He concludes that "what was needed was not so much a 'correct' conceptualization as one that could be understood by all the participants and that would facilitate action rather than paralyze it." (Simon, 1981, p. 166) What he recognized to be needed, in other words, was a process of deliberation among the participants to reach a common understanding, not some formally rigorous representation framework. Here, Simon proposed a series of methodological approaches to issues of social planning, but these were strikingly less formal than the tools he had proposed for well-structured domains. Significantly, he did not discuss the possible implementation of AI programs that might be able to support these methods of social planning.

From expert systems to critiquing

It seems clear that planning and design problems cannot be solved by means of automated methods without the active involvement of humans. Whether one thinks of Alexander's references to intuition, Rittel's insistence on the role of personal interests, Schoen's emphasis on tacit knowledge, or Snodgrass and Coyne's focus on interpretation, one finds the essence of designing in skills that are distinctively human. These skills are to be contrasted with the *modus operandi* of computer programs. During the past decade, AI research has begun to explore ways of supporting human expertise with computer systems that preserve a central role for people. A step in this direction can be seen in the shift from autonomous expert system programs to "expert critiquing systems."

In his survey of expert critiquing systems, Barry Silverman defines the term "critic" as a computer program that critiques human-generated solutions. Thus, rather than the program coming up on its own with a solution by following a set of rules that have been gleaned from domain experts, a critic program responds to a solution proposed by a human user of the program. Take, for instance, an expert system such as Simon discussed for playing chess. It would operate by accepting as input a board position and responding with an optimal move. A chess critic, by comparison, would allow a human user to make a move in response to the board position and would then critique that move. The critic might say that the proposed move violated the rules of chess, or that it put the player in some danger, or that it missed the opportunity for some better move. Most often, the critic would probably be silent and let the human continue to play uninterrupted. The idea of using critics is to allow human intuition to guide the solution process -- recognizing the appropriate role of the human -- while at the same time bringing to bear the computer's ability to recall facts, rules and constraints which the person might easily have forgotten.

As Silverman's presentation makes clear, expert critiquing systems of his style are a straight-forward modification of expert systems. They require the same ability of the computer to solve the problem, but merely delay the announcement of the computer's solution until the user has had a chance to try:

> The conversion from an expert system to a critiquing process primarily involved adding a differential analyzer that would: suppress the expert system's diagnosis until after the user had also input his or her own diagnosis (the machine would request that input), compare its diagnosis to that of the human user, and determine if the human deviated significantly enough from the machine's ("optimal") diagnosis and plan, to warrant interrupting the human to explain the problem it had uncovered. (Silverman, 1992, p. 111)

This approach can be effective in simple, well-defined domains which can be captured in a number of explicit rules or look-up tables. Spelling checkers for word processors can be viewed as a particularly successful example. Perhaps the best application is intelligent tutoring programs, where the user is not likely to be aware of all the rules of a domain which can be formulated in expert system rule bases. Most AI systems are really only "intelligent" compared to novices who are learning the basic rules, not to domain experts whose skills far exceed the realm of rules.

As the name suggests, critics can represent a first step in a paradigm shift toward the model of critiquing as a dialogue process. Consider the analogy to the human critic in a design studio, where a student does some work and then the critic responds based on principles that may have been violated or alternatives that were overlooked. In line with this analogy, Silverman claims critiquing should be a two-way, interactive, communicating, view-sharing process. Unfortunately, when one looks at the implementation details he proposes, this dialogue reduces at best to a limited user model in terms of which the program's explanatory output is adjusted to the represented skill level of the user. In other words, the program somehow classifies the user (perhaps by asking the user to select a skill level) and then prints out the text that had been programmed as an explanation for the current "user deviation" for that level user. This is scarcely an argumentative process in Rittel's sense or a dialogue in the hermeneutic sense.

In fact, much of the work Silverman presents is still very much in the rationalist tradition. In his approach, critic systems require that the domain be well-defined in terms of the following criteria: explicit rules can be specified for each type of wrong answer; the rules for assessing user solutions are objective; only one or two possible correct solutions exist for each task; and subtasks can be critiqued independently of each other. Silverman's own contribution to the theory of the critic approach is to emphasize the importance of clarity (a watchword of rationalism since Descartes). The first thing that critics should do in his opinion is to eliminate ambiguity. "Ambiguous statements which have more than one meaning cannot be clearly confirmed logically," he warns, "nor can they be completely disproven empirically. They may be true according to some interpretations." (p. 107) Although Silverman's critics have introduced people back into the problem solving loop, they have not opened the loop wide enough to permit true dialogue among competing and ambiguous interpretations.

Design environments for cooperative problem solving

In his survey of critic mechanisms, Silverman contrasts pure expert critic systems with the approach of "Colorado critics", which are "embedded in other, high-function, complex

environments" and which are part of research on "a comprehension-centered theory of human-computer interaction." (p. 112) These embedded critics represent a different attitude toward the whole relationship of computer system to user and toward the objectivity of expert knowledge. Where Silverman's expert critiquing systems were "differential critics" capable of arriving at their own solutions and then evaluating a user's solution based on its differences, embedded critics are "analytic critics" which analyze a user-generated solution in terms of a particular rule or feature. These critics are conceived as part of a "cooperative problem solving" system in which human and computer work together to iteratively generate, critique and refine a solution.

The Colorado critics are embedded within "design environments" like Janus (Fischer et al, 1993). Janus is a knowledge-based system to support the layout of kitchen floorplans. Knowledge about kitchen design is encoded in the various components of Janus' multi-faceted architecture (see Figure 2): A *palette* of kitchen appliances provides a graphical construction kit of parts for visually representing a layout. Sets of *critics* embody rules of thumb about the placement of these appliances, such as that the stove should be near but not next to the refrigerator. A *specification* checklist prioritizes client concerns, like whether the kitchen should be child-proof. An *issue-base* contains discussions of rationale for kitchen design, including deliberation related to the critic rules. There is also a *catalog* of past kitchen designs, which can be used as starting points for new designs or illustrations of abstract rules.

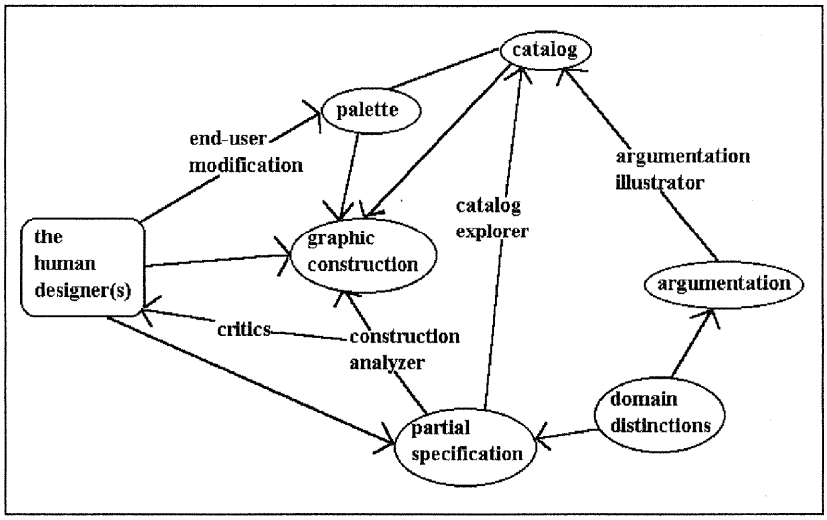


Figure 2. The multi-faceted architecture of Janus. The major components of the system are shown in ellipses; they each use a different data representation. Other components are used to bridge between these representations. Designers alternate between problem framing (altering the partial specification) and problem solving (altering the graphic construction).

Phidias is a related design environment for kitchen design (McCall, et al, 1990). It differs from Janus in developing the construction kit more fully in the direction of a CAD system, so that a designer can develop 2-D and eventually 3-D drawings rather than simply arranging fixed icons.

It also elaborates the issue-base into a more flexible Issue-Based Information System incorporating a primitive query language for navigating through the rationale hypertext. In place of Janus' critics, Phidias has triggers. These provide advice on selection and placement of appliances in advance, rather than critiquing user placements of appliances which violate rules of good design.

Both Janus and Phidias are based on the design methodology of Alexander, Rittel and Schoen. More meaningfully than Silverman, they put people in charge of the designing, so that human intuitions, interests and skills can come into play fully. Rittel's focus on argumentation is implemented in the issue-bases of these systems. Their integration of graphical construction with this textual deliberation is seen by the developers of Janus and Phidias as a way of operationalizing Schoen's theory of reflection-in-action or breakdown-and-repair, in which designers construct, observe, reflect, and respond.

Despite the progress that systems like Janus represent in meeting the needs of designers, they still fall short of supporting interpretation. Consider the forms of knowledge in such a system. The paradigm is still that the programmer who built the system obtains pieces of knowledge from books and domain experts, and enters it all in the system. Users benefit from being guided by the knowledgeable system. When designers want to, they can also explore the rationale for critic rules and defined characteristics of the standard appliances. But the bulk of the knowledge exists independently of the personal concerns of the user or the specifics of the task at hand. Recently, an "end-user modification" component has been created for Janus. (Fischer & Girgensohn, 1990) This allows users to add new appliances (e.g., when microwaves become popular they can be added to the palette) and to modify existing definitions and critic rules. However, this is not intended as a mechanism for continual redesigning of components under alternative interpretations; nor does it support multiple simultaneous definitions for different users or different interpretations.

The reliance on standardized components and relatively non-controversial rules of thumb in Janus may work in the realm of kitchen design because this domain is, in fact, well-defined and well-understood. At least if one limits ones concerns to the layout of appliances, there is a rather limited list of primitives (stove, sink, cabinet, etc.). They come in standard sizes and raise few issues for the designer. By ignoring issues of aesthetics, sociability and architectural interactions with the rest of the building, Janus is free to concentrate on rules that are independent of the interpretive perspectives of designers or their clients. For instance, the implemented critics have to do with distances between appliances, the size of the work triangle, the placement of the sink under a window, or the separation of the stove from the refrigerator. The approach of the Colorado critics may need to be extended for domains which are less well understood.

Lunar habitat design: an exploratory domain

Lunar habitat design is not a task for which one could expect to interview an expert and come up with a set of formal rules and elements to define a comprehensive system of knowledge. Workers in this field are attempting to explore a new domain and to begin to map out the potential problem space. A goal of researchers is to sketch in parametric curves that would

indicate how designs have to change depending on such parameters as number of astronauts, length of mission duration, or payload delivery capacity. (Cf., e.g., Design Edge, 1990; Moore, et.al., 1991; Kazmierski, et al, 1992) But even the most important parameters remain undefined and open to interpretation and debate. For instance, few NASA guidelines cover privacy issues, even though this is an important concern of thoughtful designers and a topic for vigorous political debate and even power struggles within NASA. (Compton, et al, 1983) Privacy is a matter of human intuition (to use Alexander's term) and subjective interpretation which resists being operationalized in the objective rules of traditional expert system approaches.

In the lunar habitat design sessions studied for the current research, privacy issues were in fact the first real concerns to surface. They structured how the designers constructed their task. Related questions of social interaction dominated questions of physical layout, indicating that social planning was necessarily a significant aspect of the designing. When the geopolitics (or solar system politics) of NASA's goals are reflected in the deliberations, the result is truly a wicked problem in Rittel's full sense. It is not just that more study is needed to formulate objective rules for the field, but that decisions necessarily involve tacit understanding of inter-personal behavior and non-propositional recognition of political interests.

For relatively unexplored domains such as lunar habitat design, efforts at designing do not seek optimal solutions within a known problem space, but begin to mark out a solution space in the first place -- as Schoen says, to construct the reality of the design situation. The most important role of computer support for such domains may be to capture the ideas that are being generated. Terms and critics which are formulated on the spot during this design exploration process are expressions of what a designer may want to pay attention to. So, for instance, the important criterion for critics is not the rigor of their computations in the sense of some rationalist engineering ideal but their ability to convey the designer's interpretive intent. The computer system as a whole should not primarily be an autonomous equation solver, but a powerful medium of external memory to empower people's creativity. An appropriate software environment for this domain would be one designed to capture new and evolving knowledge, rather than one which simply manipulates predefined knowledge representations and systems of production rules.

A high-tech design goes through many stages of development, involving different design teams. Architects, designers, a variety of engineers, and administrators all work on the designs from their own viewpoints. Successful designs are sent to other contractors around the country for detailing, mock-up, testing and construction. At each stage, the design is modified, based on people's understanding of the design and its rationale. If a creative design concept is to survive this argumentative process -- with tight cost, weight and volume constraints at every stage -- strong rationale must be communicated; a schematic diagram or a pretty picture will not suffice. In fact, a typical product of lunar habitat design consists of a small booklet predominated by textual explanations of rationale, not just detailed drawings. The important role that rationale plays in this extended design process should motivate designers to document their reasoning and interpretation more than they would in a domain like kitchen design.

Because designers lack personal experience living in lunar habitats, knowledge embodied in previous related designs (including Skylab, the Shuttle, Space Station Freedom, previous trips to the moon) is invaluable. Old designs are re-used extensively. To the degree that design rationale

of the old designs has been captured and augmented by subsequent experience, it is vitally important. Consequently, it is likely that design rationale will increasingly become an integral part of design. This should add tremendous power for practitioners who take it seriously and those who use computer tools that support rationale capture. Such a development represents a significant break with the tradition of CAD programs, which are purely graphical and embody very little semantics. However, it has impressive precedence in other fields like science, mathematics and philosophy, where written theories, proofs and arguments have been refined through processes of public critique and have grown into extensive bases of shared knowledge and accumulated commentary impossible in non-literate cultures.

The need for computer support of lunar habitat design was originally suggested by the sheer volume (complexity) of knowledge required -- far more than people could maintain in their heads or even locate easily in manuals. There are voluminous sets of NASA regulations for all Man-In-Space designs, ergonomic standards, and specific project contractual obligations which must be adhered to by designs. But the complexity of lunar habitat design is not just a matter of the volume of information. Requirements, components and rationale all have to be reinterpreted within the *Gestalt* of the evolving design. This is an application realm in which, for instance, most physical components require some degree of customization. Because of gravitational or volumetric considerations, one cannot simply select a stock sink or bed from a catalog, Even pumps and fans must be re-thought. Furthermore, there are many design interactions among components that are placed close together -- partially because space is at a premium and also because things must work together to form a coherent environment for habitation. This means that design of a given part is very much situated in its context, in terms of neighboring components (e.g., sound buffers), design concerns (privacy), and projected usage issues (traffic flow). The computer representation of the design must function as the unique world in which representations of all the components and their relationships are appropriately situated so that design can take place effectively. One wants to start from existing components, but one then needs to be able to modify them freely to account for differences in the lunar setting. So representing standard parts with schematic icons or fixed items from a palette is inadequate.

Elements of lunar habitats should be similar to familiar products to facilitate manufacture and to give astronauts a sense of being at home, but they must also be different to meet the severe constraints of their context. This means that models and rules of thumb must be searched for in many other domains (houses, submarines, Antarctic labs) and then applied to the lunar setting. Hermeneutic theory teaches that application is not a mechanical process; it must be done by the creative and synthetic minds of humans, with computer systems merely presenting the relevant elements. Even the determination of what might be relevant must involve the human designer, for this is also very much a matter of interpretation based on a deep understanding of the semantics involved.

To support the subtlety of communication between the computer system and its users, the users should be able to develop a language that operationalizes their evolving interpretations in ways which can be used by the software. At the same time, the development of a language for interpretation can provide a basis for shared understanding among groups of designers, even if they are not working together at the same time or place. For instance, a designer who is considering an old design for adaptation into a new project can learn about the old design through

the language which was developed with it -- including the formulations of critics specific to that design. Providing some support for collaborative work among groups is particularly important in this domain because of the way each successful design must undergo the scrutiny of many teams. Generally, the only communication between these teams is the design document itself. To further mutual understanding, it is desirable that the design include effective documentation of the interpretive stance behind the rationale.

Lunar habitat design is not a task for one person sitting at a computer. It proceeds through the work of teams of teams, each viewing the common product through their own perspective. The essential communication is not that between a human and a computer (the model underlying Janus and Phidias), but among the design teams. What a computer system like Hermes can do is to provide an electronic medium to support this communication. It can do that by facilitating the development of a shared language of design interpretation and by providing a mechanism for the creation and sharing of interpreted designs defined using that language.

III. Hermes: a Computational Medium

A scenario using Hermes

Hermes is a prototype software system crafted to support the work of designers engaged in tasks like lunar habitat design. It is intended to support human interpretation by keeping control over all aspects of the design process in the hands of the system users and providing ways for them to articulate, share, visualize and store their interpretive perspectives of the emerging design. It is also conceived as a response to the requirement of design methodology that the representations in terms of which a design is interpreted emerge out of the design process itself.

The following scenario of Hermes in use is meant to give an initial feel for what it is like to design and to construct interpretations of designs using Hermes. Emphasis is placed on some of the variety of ways in which Hermes users can define and modify the computational representations of the design knowledge they create or discover. For practical reasons, the scenario is kept simple. Some of the procedures and underlying mechanisms will be described more fully in subsequent pages.

The task for the scenario is to design a lunar habitat for the mission which has already been described. A design team consisting of Adam, Betty and Chan has just met to discuss a preliminary sketch by Adam. They have focused on the issue of privacy and agreed that for many reasons it is problematic that Adam's design has a common area opening up into the sleep area and the hygiene area. Betty agreed to define and document the problem more precisely using Hermes' language. Then she will pass the problem on to Chan, who is to propose a possible solution prior to the next group meeting.

For the sketch he had prepared before the meeting, Adam had defined a work context for himself in Hermes and within it had created a lunar habitat drawing consisting of three component areas: a sleep area containing four bunks along the walls; a common area consisting of galley and wardroom subcomponents; a hygiene area containing a shower and a WC. (See Figure 3.) Adam selected the bunks, galley, wardroom, shower and WC from a palette of components from previous lunar design projects. He added his three habitat areas to the palette and added his overall habitation drawing to the catalog of designs. While his ideas were fresh in his mind, he entered his rationale for the design, expanding the design rationale associated with the components that he had copied into his work context.

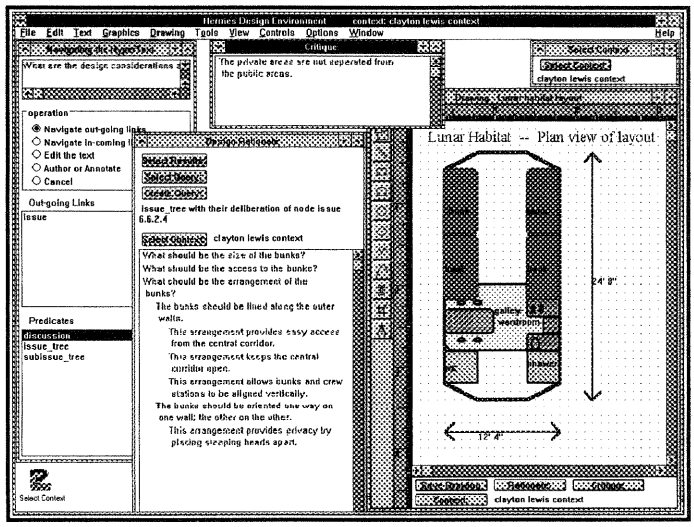


Figure 3. The Hermes screen while Betty is working. She has the drawing that Adam prepared. The predicate dialog box shows a predicate she has defined in the Hermes language, using a type selected from the list of types. The critique window shows the result of evaluating her new critic. [N.b.: this figure must be revised to match the caption and text.]

Adam has created a graphic representation of a design. Sometimes this kind of drawing appears to be the primary product of the design process. However, if one videotapes designers at work and has them verbalize, then one discovers that the drawings are but shorthand reminders (external memories) of extensive consideration of issues and refinement of terms. In traditional (non-computer-based) settings, much of this is communicated through gestures, abbreviated utterances and informal comments -- little of it is captured explicitly in documents. Hermes captures these elements of design interpretation in formats that allow the computer to provide powerful computational supports which never before existed. This transcendence of traditional means (Ehn, 1988) comes at the cost of making series of explicit definitions, as can be seen in Betty's role in the scenario. (Betty's series of definitions are described below and are summarized in Figure 4.)

Betty defines a new work context for herself which inherits Adam's context, so she will have access to Adam's work (design, palette, rationale, etc.) and will be able to modify her version of his work without affecting the original version in his context. She selects his design from the catalog and decides to start by defining privacy values for the three component areas of the habitat. First she defines a relationship named privacy values by adding this phrase to the list of relationships already defined in Hermes. She then assigns numeric privacy values to Adam's three components by sliding her mouse along a scale and linking the value selected to its corresponding component in the lunar habitat design. Using this definition of privacy values, Betty can now define public areas as components which have privacy values which are less than 3 and private areas as components which have privacy values which are more than 7. She does this using the Hermes interface for defining predicates in the language.

Next Betty wants to add a specification concerning privacy to the list of partial specifications for the design. She defines a new issue (which she names the privacy specification) in the design rationale: "Is privacy important?" She accepts this new issue in the issue-base (assigning it an answer of true) and adds it to the list of specifications. She also defines two phrases in the Hermes language: if privacy is important (her name for the conditional phrase which she creates: if there are answers of the privacy specification which are true) and the privacy message (her name for the message she writes: "The public areas are not separated from the private areas"). Now she is ready to use these phrases to define the critic which will analyze the drawing in terms of the privacy values of its components. Privacy critic is defined by Betty as a query in the Hermes language: Display the privacy message, if private areas are near public areas, if privacy is important. When this critic fires, the query will evaluate the answer to the privacy specification to see if privacy is important. Then it will check that there is at least a minimal distance between each private area in the drawing and each public area. If these conditions are true, then the privacy message will be displayed on the computer screen.

Betty adds the privacy critic to the list of critics and has the lunar habitat drawing critiqued by them. She reads the privacy message which appears and decides it is not very informative. So she modifies the critic to display rationale in addition to the message. She adds some answers to the privacy specification issue and several arguments to these answers. Then she defines a computation named debate: answers with their arguments. Now when it is evaluated the critic displays the privacy message followed by the debate of the privacy specification, as shown in Figure 3. Betty is happy at this point. She feels this gives an operational definition of privacy which can be used to test variations on lunar habitat designs.

relationship:	**privacy values**
predicate:	**public areas = components which have privacy values which are less than 3.**
predicate:	**private areas = components which have privacy values which are more than 7.**
issue node:	**the privacy specification = "Is privacy important ?"**
answer node:	**= true.**
conditional phrase:	**is privacy is important = if there are answers of the privacy specification which are true.**
text node:	**the privacy message = "The public areas are not separated from the private areas."**
critic query:	**privacy critic = Display the privacy message, if private areas are near public areas, if privacy is important.**

Figure 4. A series of terms that Betty defines, culminating in the privacy critic.

Chan takes over Betty's work by inheriting her context in his. He sees the critic message and realizes that the sleep and common areas must be separated. He has an idea. He rotates the bunks into a pinwheel arrangement. That frees up space to insert storage cabinets and a doorway to isolate the sleep area in terms of sound and light from the common area. Chan adds his thoughts on privacy to the issue-base that Betty began. To cause the whole extended discussion

to show up, he modifies debate to be issues and answers and arguments with their debate.

Now Chan tests his design and the critic still fires. He decides to separate the hygiene area from the common area by building a five inch wall between them. The wall can hold the plumbing and water reserves for the shower and WC, so those rooms can be made five inches narrower. A door is placed in the wall, providing a private dressing room. The wall can also contain equipment for dust control, so that astronauts entering the habitat through the air lock at that end of the habitat can refresh themselves in privacy before entering the common area, and moon dust they brought in will be caught more effectively. (Chan's work is shown in Figure 5.)

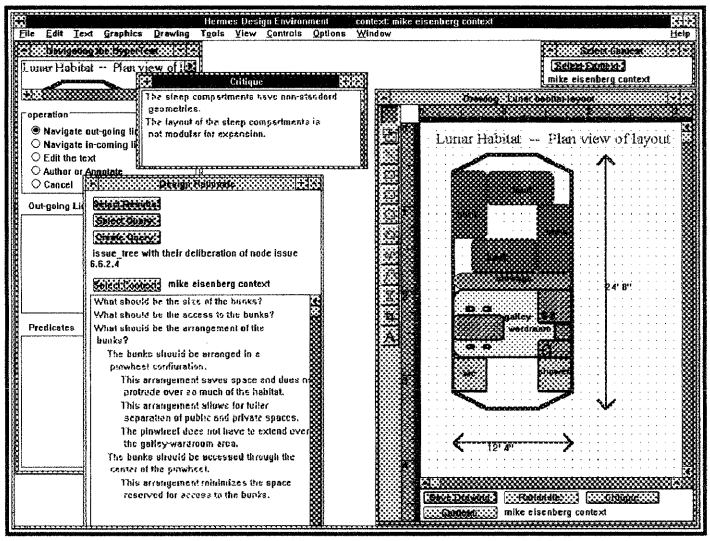


Figure 5. The Hermes screen while Chan is working. The context control shows how he defined his new context to inherit from Betty and from other contexts. The drawing has been revised by Chan in response to the critique shown in Figure 3. In the query window, Chan has tested his new definition of "debate". The navigate dialog allows him to browse through the rationale in the query window. The result of the critic with the revised drawing shows that no problems were found. [N.b.: this figure must be revised to match the caption and text.]

Chan's revised design meets Betty's critic tests. His rationale has been entered for others to review before the meeting and to amend at the meeting. Meanwhile, anyone can set up a work context which inherits Chan's and try out variations on his design, perhaps modifying some of the definitions that Betty made or altering the graphical components.

Interpretation in design: defining the representation

A closer look at what took place in the preceding scenario will show how designers can define their interpretations in Hermes, how these are represented in the computer system, and how

Hermes can use this information computationally. The design team in the scenario interprets their task as one of providing both public and private space in the habitat. In order to distinguish public spaces from private ones, Adam interprets his concept of the habitat layout as consisting of three functional areas: a private sleep section, a public common space, and a private hygiene component. Adam sketches in these areas using Hermes' graphical editor.

Betty's task is to explicate the design group's tacit understanding of privacy, which is based on their comprehension of the semantics of the English word, their personal experiences of having wanted to be alone, and their knowledge of various architectural examples of private spaces. She needs to reinterpret this concept in a way which can be used by the computer computationally. She decides to do this by defining it as a numeric value from 1 to 9 which can be associated with a graphical component. Objects with privacy values close to 9 will be considered private; those near the other end of the scale public. One might well wonder what the number 9 has to do with the meaning of "privacy". The point is that human judgment is still needed to make the connection between a graphic component and its privacy value. But once this judgment has been made and embodied in the assignment of a value, the computer system can use the value in its manipulations. This approach does not exclude the possibility that in some cases it may be possible to formulate an algorithm for the computer to assign privacy values directly, based perhaps on the topology of room layouts. But such an algorithm is not necessarily available, and would at any rate require a human judgment to accept its calculation as equivalent to the meaning of privacy.

Betty develops her ideas in Hermes, a computer-based design environment which has no built-in knowledge related to privacy. Instead, it provides a language in which new concepts like privacy can be defined and referred to. The language is integrated throughout Hermes, so that it can apply to the graphic drawings or to navigation through the issue-base of rationale, and can be used in any function within the software system. The language provides for the definition of several kinds of phrases, which can be built up from simple relationships like "privacy value" to form arbitrarily complex clauses. Betty's central task is to define a critic in the Hermes language which will check the privacy values of graphical components in the habitat drawing and will display a message if a component with a low privacy value is near a component with a high privacy value.

To build her critic, Betty first needs to define a number of phrases: a term for public areas, a term for private areas, a message for the critic to display, and a clause to check what components are near what components. (See Figure 4.) In addition, Betty wants to add a condition to the specification for the habitat stating that privacy is a concern. She will make her new critic conditional upon this specification being accepted in the design, so the critic will only be active in designs where privacy is an issue. These are the steps Betty goes through in her part of the scenario.

To give a feel for the use of the Hermes language, the construction of her first phrase -- the predicate for public areas: components which have privacy values which are less than 3 -- will be described in some detail. First, Betty indicates that she wants to define a new predicate. A form appears on the computer screen asking her to name the predicate and select its format. She types in the name, "public areas". Of the dozen alternative syntactical formats listed, Betty

chooses the one that says Relationship which Filter. This means her predicate will consist of two subclauses, a Relationship and a Filter, joined by the word "which". Hermes presents a list of all defined Relationships and Betty selects "components". Then Hermes presents a list of formats for Filters. Betty selects the syntax: have Relationship which Filter. This time, for Relationship, she selects "privacy values" and for Filter, "are less than Number." Finally, for Number she enters "3". Hermes displays the desired clause, consisting of the selected terms joined together in the nested syntactical forms. (Figure 6 shows the internal structure of this phrase.)

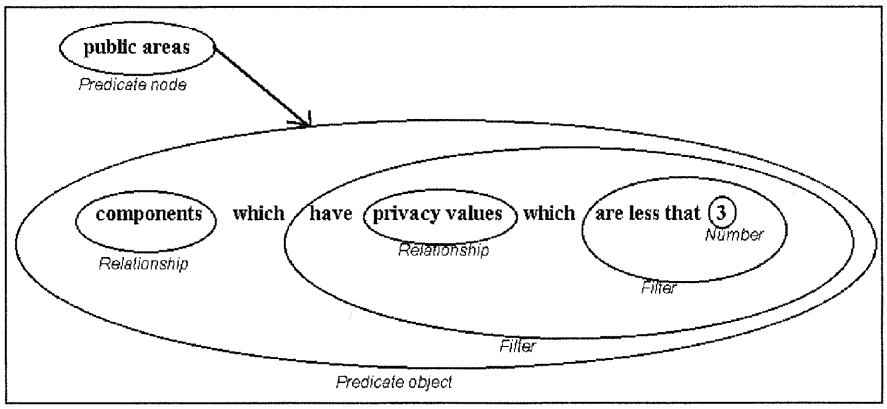


Figure 6. The syntax of Betty's predicate "public areas". It is a nested structure defined by syntax options of the Hermes language, simple relationships (link types defined by Adam or Betty) and system primitives (like the number 3).

The process just described makes much more sense when conducted on the computer than when read as a narrative. Nevertheless, even in the user-friendly Hermes system it is not a trivial undertaking for people to construct critics and their constituent clauses. It is assumed that Betty would have received some training in the language and would have gradually learned how to use it through seeing examples of existing clauses, trying to modify them, and experimenting with simple exercises. The Hermes language is, after all, a powerful programming language, and Betty is a professional who is willing to invest considerable cognitive effort in learning to use valuable tools of her trade. On the other hand, the Hermes language avoids many of the considerations which make most computer languages hard for non-programmers to understand. Also, when thoughtfully constructed, clauses in the language sound English-like, which helps people to remember what they do.

A key feature of the language is that virtually all the vocabulary -- except for the syntactic sugar which makes it stick together: words like which, have, are -- is defined by users. Thus, phrases tend to consist mostly of words that are commonly used by workers in the domain. This is part of what it means in Hermes for users to create the computer representation that they will be using. Just as Adam creates the graphical representation of his habitat design, including its

break-down into three major subcomponents, Betty creates the phrases in terms of which her team will interpret the design and the computer will analyze it.

The mechanisms of Hermes: overcoming the limits of representation through the power of representation

The key to understanding how Hermes works is to appreciate its hypermedia structure. The term hypermedia means nothing more than that *nodes* which can have contents of various kinds are linked together in a network by *links*. For instance, Betty has created a new node named the privacy specification. It is of type issue and has the following text for its content: "Is privacy important?" In figure 7, nodes are represented by ellipses (which can contain a name) and links are represented by arrows (which may be labelled with a type). Note that the content of a node is separate from the node itself in Hermes, and the two parts are connected by links the same way that different nodes may be connected. A link of type answer goes from the issue node to an unnamed node whose content is the true/false value true.

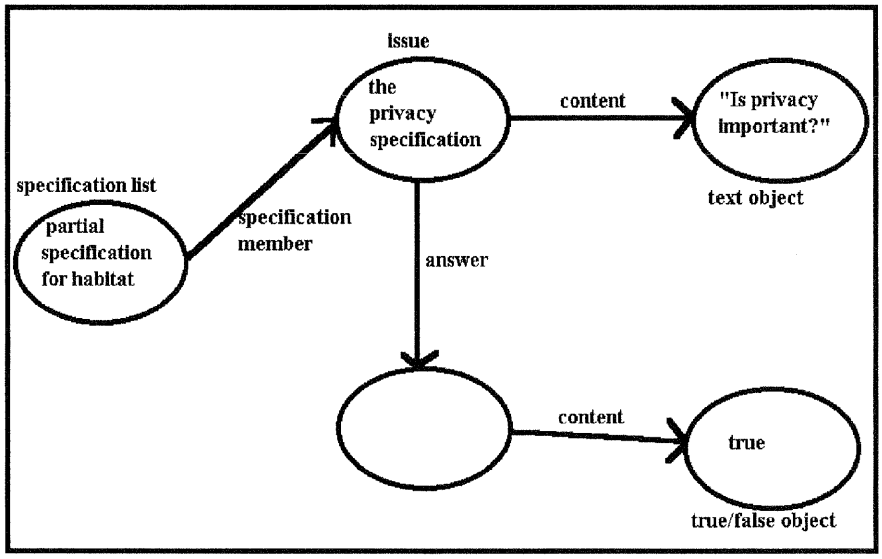


Figure 7. Part of the hypermedia network. A link of type "answer" connects an issue node to its answer node. Links of type "content" connect these nodes to their contents. A named partial specification list is maintained by linking a node with the name to nodes for each of its members.

These nodes and the links joining them are part of the hypermedia network that includes all the data in Hermes. In this sense, Hermes is a database management system that stores, organizes and retrieves multimedia data: text, numbers, true/false conditionals, line drawings, scanned images, phrases in the Hermes language, data lists. It could be extended to include sounds, video clips and animation sequences as well. Each medium has procedural methods associated with it. For instance, each medium has its own display method, so that text objects are displayed as lines

of characters in certain fonts and sizes, which wrap to the next line when they reach the right margin; numbers are displayed in certain decimal formats; drawings are displayed graphically. The structure of the hypermedia in Hermes, which (as will be discussed in the next subsection) incorporates the Hermes language and the context mechanism, determines the power of the database system. The Hermes language acts as a database query language to retrieve information by navigating through the hypermedia links, collecting nodes whose contents are to be displayed. The interpretive contexts provide a means for organizing the data into different views and versions.

The graphics system in Hermes is based on the hypermedia structure. The habitat that Adam defined, for instance, has as its content the three component areas. These have as their content various other drawings which Adam choose from a palette of stored graphics. In Hermes, graphics can be composed of arbitrarily complex networks of subcomponents linked together, ultimately formed out of simple rectangles and other polygons. Because the language is part of the hypermedia structure, graphics can be composed using predicates and queries as well as direct links. Thus, the content of a graphic object can be dependent upon some computation or upon what is currently present in the database. Of course, the context mechanism applies to the graphics as well, so that the habitat drawing is displayed differently in Adam's and Chan's contexts. As explained below, contexts provide a powerful mechanism for virtual copying, so that multiple copies of a given graphic can be stored efficiently and still be modified individually.

The fact that all data is integrated in the hypermedia network allows many system functions to be implemented simply. For instance, the palette, catalog, list of active critics, and specification forms -- which each require a complex software component in the multifaceted architecture of Janus -- can be defined as sets of links from particular nodes. In Figure 7, a node named partial specification for habitat is linked to an issue node named the privacy specification. It could be linked to numerous other issues as well. This linkage defines a design specification. Any use of specifications can access the first of these nodes and then browse through its links. Moreover, there can be more than one specification list defined at a time, distinguished by their names, and the contents of the specification lists can be different in different interpretive contexts. Similarly, palettes of component drawings, catalogs of habitat drawings, and lists of relevant critics can be defined and maintained by designers to meet their needs. Simple supports for managing these lists are provided in the Hermes user interface. Hypermedia also provides a natural implementation for the triggering of argumentation on selection of palette items and their placement, which requires a special mechanism in Phidias: in Hermes this can be accomplished by users defining and navigating standard links from graphical objects to text entries in the issue-base. The hypermedia structure of Hermes is the basis for an integrated software architecture which not only simplifies the programming of additional functionality, but facilitates functional extensibility by individual and group users.

The integrating data structure for Hermes is, in several ways, unique even for hypermedia:

(1) The *content of a node is separated from the named node* and connected to it by a link. This separation is useful for implementing the context mechanism and other forms of inheritance.

(2) The *granularity of text and graphical nodes is finer* than in most hypermedia systems. This allows reformatting of views of data based on interpretive perspectives.

(3) The nodes and links are part of a larger *object-oriented system of data types*, including the elements of the language and nodes of the various media. This allows the definition of inter-dependencies among these various kinds of objects.
(4) The *links* contain computational information concerning the *contexts* in which the nodes they connect are to be displayed. They also store knowledge about the position and attributes of graphical nodes that they are linked to.
(5) The *language* is embedded in the *nodes* of the hypermedia. This means that predicates, conditional clauses, and queries in the language can be used to determine the content of nodes and links.

(1) The separation of nodes from their content was already illustrated in Figure 7. An advantage of this is that a single named object (a node) can have multiple contents, for instance in different interpretive contexts or under different conditions. The mechanisms for this will be discussed in the following paragraphs. This provides a balance of flexibility (in contents) and consistency (in names). For instance, different designers might word the privacy specification issue differently or provide different contents for its answer. However, for everyone using the issue-base, there would be an issue called "the privacy specification" and an answer to it. So the conditional phrase which Betty defined -- if there are answers of the privacy specification which are true -- would be a meaningful, well-defined condition when evaluated in any context with any combination of answer contents.

Similarly, the independence of named nodes from their content is useful for defining inheritance hierarchies of nodes. Suppose you wanted to distinguish three related types of answers: general answers, short answers, and cryptic answers, where all cryptic answers are also considered short answers and all short answers are also considered general answers. Then, a condition that specified, if there are short answers ..., would have to check for short and cryptic answers but not general ones. Hermes implements the definition of this form of inheritance with links from a node containing the name of one type to a node containing the name of the type it inherits from. When evaluating a phrase that contains a type name, Hermes can check the inherits from links going in and out of the node with that type name. As will be seen in Figure 8, a type like debate can have different contents in different contexts -- for instance, those defined by Betty and Chan -- but always have the same node-to-node link structure. Thus, if debate were part of an inheritance hierarchy of types, the inheritance links could be defined independently of the defining contents. Of course, if one wants to redefine the inheritance in different contexts, that can also be done just as the annotation link discussed in paragraph (4) below is redefined. But even so, the context dependencies of the content definitions and those of the inheritance structure are in general independent.

The interpretive contexts (also referred to as work contexts, perspectives, versions or virtual copying contexts) form an inheritance network. For example, Chan's context inherits Betty's, which inherits Adam's. This means that when Chan is working in his context anything defined in Betty's context will be defined the same in his context, unless he has deleted or modified that definition. Similarly, anything defined in Adam's context will be defined the same in Chan's context, unless either he or Betty has deleted or modified it. To implement this in Hermes means that whenever something like the debate type is evaluated in Chan's context, the computer may need to check for its definition in all contexts that Chan's context inherits from. The hierarchy of

these inherited contexts is maintained as a link structure in the hypermedia, just like the forms of inheritance discussed above. The only difference is that navigation through the context network is highly optimized and ignores the currently active context. Because contexts are defined within the hypermedia, few special mechanisms are required to handle them. Also, they can be easily linked to other information, such as free-form annotations, their creation date or security passwords.

(2) Because Hermes needs to display information in accordance with interpretations that are not pre-defined but are defined by the user, all displays must be computed dynamically. This is done with queries as opposed to the page-based approach of most hypertext systems. In a system like HyperCard, a presentation of design rationale might contain a page full of issues. Embedded with an issue would be a button for its justification. Clicking on that button would bring up another page of text presenting the justification. Similarly, in Janus a page of design rationale would contain highlighted terms; clicking on one of them would display information about that term, allowing one to browse through pages of related textual information. In Hermes, however, the justification must be recomputed based on the current interpretation. This is done by executing a query based on the information desired (e.g., justifications of an answer to a certain issue) and based on the definition of the current interpretive context. The results of the query are then displayed, in place of a pre-formatted page. This approach was adopted from the Phidias design environment, which featured a primitive query language for allowing the user to structure the textual displays. (McCall, 1989) The fact that in this approach design rationale is generally stored at the granularity of sentences rather than pages means that it can be modified by changing or adding short sentences, or by modifying the definition of the query, as Chan did in the Hermes scenario.

(3) In Hermes, the hypermedia is inseparable from the Hermes disclosure language. Hermes is implemented as an object-oriented system, built around a system of approximately one hundred object classes that define the language options, the various media and the hypermedia elements. Many of these classes are defined in terms of each other. For instance, there are three classes of nodes: simple nodes, conditional nodes and virtual structures. Simple nodes can have contents (which can be instances of any of the hundred classes of objects) and can be labelled with a user-defined kind. Conditional nodes have, in addition, a true/false clause. If their condition is true, conditional nodes have their normal content just like simple nodes; if it is false, they behave as if they had no content. Virtual structures are like simple nodes except that in place of their content they have a query to be evaluated. Their effective content is the result of the query. The query is, of course, an object defined in the language. The true/false clause is a media object. True/false clauses can have primitive boolean values of true or false, or they can be computed from more complex options which, for instance, compare two numeric values or check if there are members of a set of nodes which satisfy a filter condition. In the last possibility mentioned, the true/false clause would be defined in terms of a subject clause of the language (which selects the set of nodes) and a filter clause (which performs one of a variety of tests on each of the selected nodes).

There are conditional links in Hermes, just as there are conditional nodes. These are links which can only be traversed if the condition evaluates to true. There is also a link analogue of virtual structures. This is the use of predicates, clauses in the Hermes language which define complex relationships or link types. For instance, if link types answers and arguments are

already defined, then Betty's definition of debate as answers with their arguments defines a predicate. If there is an issue in the hypermedia issue-base which has answer links to nodes which in turn have argument links to other nodes, then evaluating debate of that issue will produce a list of the answer nodes, with sublists of their argument nodes. In this way, a predicate defines a virtual link computation, linking the original issue node with a set of other nodes to which it may or may not have been directly linked.

There are many other examples of language, media and hypertext classes which are defined in terms of each other. The integration of the language with the hypermedia will be discussed further in the following subsection. This integration gives Hermes its flexibility and power. In particular, the fact that mechanisms like type inheritance and conditional nodes are part of the hypermedia database and that the contents of this data is definable and modifiable by the system users gives designers who use Hermes the ability to develop and control sophisticated design representations. By treating all design knowledge as interpretations stored in a shareable, programmable hypermedia rather than as predefined representations, Hermes overcomes the central problem of computational representations of knowledge by allowing then to evolve with the design solution. In a sense, Hermes provides a rich system for representing design representations, so that designers can design their own representations of problems as they explore interesting solutions. In this way, Hermes uses the power of computational representations to overcome the limitations of *a priori* knowledge representations.

Shareable, programmable hypermedia

(4) In the scenario, Chan redefines Betty's "debate" predicate to be issues and answers and arguments with their debate. To evaluate this predicate applied to an issue node, the system will make a list of all nodes connected to the original issue by issue, answer, and argument links. Then it will form sublists of their debate. But debate is once again (recursively) defined as this predicate, so the system will collect the nodes connected to each node in the first list by issue, answer, and argument links. If the issue-base has a rich network of nodes connected by these links, this process will continue for some time, building up a list with sublists, with sublists, etc. until it reaches the ends of the tree of links of these types. (If there are cycles in the network, the system makes sure it does not go around the links repeatedly but eventually completes its computation.) An arbitrarily large tree of nodes will be displayed as the debate of the original issue.

The context mechanism which allows Chan to redefine Betty's predicate is implemented in the hypermedia links. Figure 8 illustrates the node for the debate predicate with content links to its two definitions. These links are labelled with information about which contexts they are associated with. When the system is in Betty's interpretive context, the top link will be navigated to provide Betty's definition of the predicate. This link was labelled "Betty" when it was created because the context named Betty was active then. When Chan redefined debate, the Chan context was active, so the link to his definition of debate was labelled "Chan". If the system has to evaluate the debate predicate while Chan's context is active, it will navigate the second link marked Chan. However, Chan's context inherits from Betty's, so the system will also navigate a link labelled Betty. To avoid this confusion and to clarify that Chan's new definition is a

redefinition of Betty's, the system must also re-label the first link to explicitly exclude its being navigated within Chan's context. This is shown in the Figure, with the first content link now being labelled Betty, but not Chan. The basic mechanism for contexts is fundamentally like that of conditional links, with minor variations to handle deletions, modifications and other special cases.

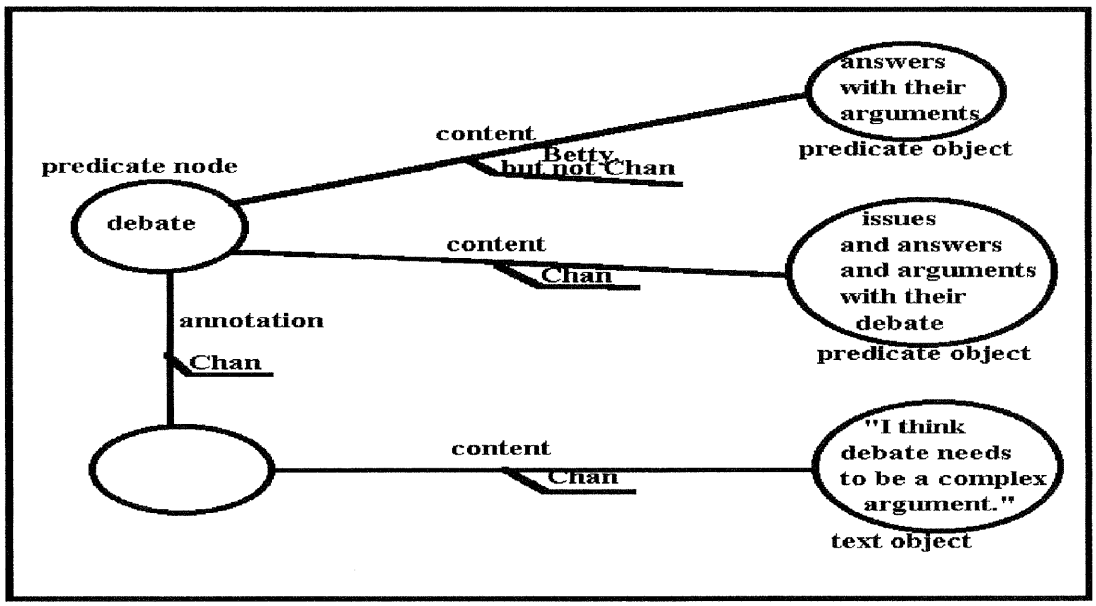


Figure 8. Chan has modified the content of the node named "debate" and annotated it with a brief explanation. The links between nodes and from nodes to their content are labelled with context information. The content of debate is one thing in Betty's context, or in any context that inherits Betty's context except in Chan's context or in any context that inherits Chan's context. It is the new content in Chan's and his descendents' contexts. The annotation only appears in Chan's and his descendents' contexts.

The context mechanism can be used in a variety of ways in Hermes. This paper has stressed its use in establishing personal and group interpretive perspectives. By defining a work context, a designer or design group can organize their definitions of graphics, rationale, specifications, critics, catalogs, palettes, triggers, predicates, filter clauses, virtual structures and queries. A set of these identified by the context in which their definitions are active can constitute, in effect, an interpretation of a design task by determining the graphical elements, the critical perspectives, the display views, the argumentative slant and the terminological framework. Inheritance of these contexts allows individuals to adopt and then modify the interpretations of other designers, of their design team and of projects in the past which were created in Hermes. This makes the hypermedia system highly shareable by its users.

The inheritance of interpretations also permits the creation of version hierarchies. A designer can develop whole networks of versions of tentative, alternative designs for a given project. Rather than consisting primarily of graphical sketches, as traditionally produced by designers, these versions would include textual rationale, interpretive critics, and so on. Versioning is a

feature that most hypermedia systems lack (Halasz, 1988), but that is clearly important for database management as well as for exploratory design.

The inheritance of contexts of design rationale was an original motivation for the Hermes project. NASA has volumes of design guidelines for manned space missions, including a general volume and one for space stations in particular. Ideally, it would be desirable to take the general volume and make a version specific for low gravity (e.g., moon and Mars), a more specific one for lunar habitats, and possibly a series of even more specific guidelines for particular types of lunar missions. Context inheritance provides a means of producing and maintaining these multiple versions consistently. Each version would inherit from the next more general one and would modify only what is not appropriate in the more specific context. Furthermore, individual design teams could annotate their own versions of the manuals when they wanted to add corporate or personal guidelines or to present reasoned arguments for exceptions to the official version. This would show that they consulted the manuals and would document why they may have justifiably deviated from the guidelines in specific instances.

A major advantage of this mechanism is that the information that is common to multiple versions is only stored in the hypermedia once. There are not multiple copies to fill computer disk space or to cause problems or inconsistency as the general rules evolve. The Hermes contexts implement a hypermedia approach to "virtual copying" or "copy-on-write" techniques that have long been used in CAD graphics and operating systems like the Mach system for NeXT computers (Fitzgerald and Rashid, 1986). By storing copying information in the links which already exist, Hermes avoids the additional mechanisms which cause considerable complication in other systems, while sharing their substantial gains in efficiency. In addition to identifying an item's context, the links store its display attributes and spatial transformations. For a text item, these attributes include its font and color. For a graphical element, scaling, rotations, and translations are computed as the links are traversed to display composite graphics.

A CAD-like use of contexts in Hermes is for storing multiple copies of a graphical palette item. Suppose that there are ten chairs in a lunar habitat design, all of the same style. Then there would be a single node for the chair in the database and ten links from the node for the habitat to that chair node. Each link would contain a different context identifier, as well as coordinate transformations for the placement and orientation of the corresponding chair instance. The context identifiers would be created automatically when the chairs were placed into the habitat; these contexts would not be named, so they would not show up on lists of contexts and be confusing to users. If the designer wanted to alter a given chair's design (its height, color, support, etc.), then just that changed part of the one chair would be copied and modified in its context, adding minimally to the usage of storage but allowing the flexibility that is excluded from other graphics systems that try to achieve this saving (Foley, et al, 1990, p. 342).

(5) The Hermes disclosure language is embedded in the nodes of the hypermedia, much as the contexts are located in the links. As already noted, a node can have as its content a query, which evaluates to a virtual structure, and a node or a link can be conditional upon a true/false clause. This means that the display of that node -- whether it is part of a graphic drawing or a piece of design rationale -- will be determined by the content of the hypermedia database at the time the display is computed. The evaluation of the queries or conditionals may in turn require the

evaluation of strings of other queries or conditionals or predicates, which may be modified by various users over time in different contexts, just as Chan revised the definition of debate, the graphic that was tested by the critic and the text that was displayed by the critic query. This makes the hypermedia very programmable by the end-user.

Some of the other benefits of storing language definitions in the hypermedia have been mentioned in passing: First, it allows definitions of clauses to be modified by context. Second, users can attach rationale directly to these definitions, to document the reasoning behind the particular definition. Third, terms can be included in inheritance hierarchies. Fourth, the definitions can be displayed by users through the normal browsing interface. Fifth, the language can be used in the definition of virtual structures. Finally, the language is thoroughly integrated with the rest of the Hermes system. These are important advantages for a system to support interpretation.

Everyday language is the ultimate medium for interpretation. Our ability to use language is what allows us to disclose things *as* certain kinds of things, and thereby to comprehend them. This is what Gadamer has in mind when he claims that "being, which can be understood, is language" (Gadamer, 1960). As Schoen noted, it is in the reflective conversation with the materials of the design situation that the artifact and the designer are both disclosed *as* what or who they are. This happens in the process of explication in which what was tacitly anticipated becomes expressed in language.

A primary stage of language use is naming. Accordingly, the Hermes system allows all objects in the design environment to be named by its user. Graphical objects in a drawing, textual statements in the rationale, critics, etc. can all be named if desired. This enables the user to refer to them in other statements, such as critics and queries, and to access annotations attached to them.

Perhaps the next most basic use of language is for categorization For instance, statements in the Hermes issue-base (or any other objects in the system) can be categorized with a type when they are created. The links connecting them are also given a type. Thus, an answer might be related to an argument via a justification relationship. Then one can request a display of all the argument statements that are related by justification links to answer statements of a given issue statement with the query, Display all justifications of answers which are arguments. Queries like this are fundamental to the ability of Hermes to support interpretation. Of course, the types themselves are created by users as are all terms and constructs of the language, for the point is to allow designers to design their own representations.

Only certain features and structures of the Hermes disclosure language have been described here. For instance, computations can also be defined by users in the language. A calculation of total private space in a lunar habitat could be expressed using a predicate for privacy, some measurements of graphical objects in the drawing, and arithmetic operations. Complex critics using such computations could be built up modularly from component definitions of predicates, filtering clauses, etc. The calculation of total private space would be named and referred to in the critic, which might check that the result of that computation was at least a certain amount per astronaut. A query could also request that all private spaces in the drawing be displayed, or

highlighted, or shown in red. (Cf. Stahl, 1991, for a fuller discussion of the language.) The language could be further expanded in many ways to implement new functions or to enhance its usability. The point for now is its core role in providing a shareable, programmable medium for supporting interpretation in design.

Conclusion: Evolving Interpretations for Design

Modern scientific knowledge was rendered practical by the medium of written language. (Donald, 1991; Norman, 1993) Writing provided an external memory for people, overcoming the limitations of human memory, especially short-term working memory. It let them put down their ideas where they and other people could view them, criticize them, and refine them. It facilitated the communication of ideas and the evolution of shared perspectives. The Hermes design environment -- named after the wing-footed Greek messenger god credited with discovering both spoken and written language -- aims to extend the medium of external memory from static paper to a highly computational medium. The idea is to allow designers to represent their project concepts, graphical forms, design rationale, and interpretive perspectives in a system which can dynamically make use of these representations to produce displays which disclose new views of the design situation for people to react to. The vision is that as Hermes is used by design teams working on related projects its store of design concepts, rationale and interpretive devises will grow into a significant computational medium for designing.

Traditional AI always sought clever representation schemes which allowed an automated system to solve the problems of well-known and narrowly-defined domains. The perspective on design methodology presented in Section I of this paper argues for a more flexible representation style which empowers the intuitive skills of humans rather than trying to replace them with algorithmic computations. Heidegger's analysis of interpretation based on pre-owning of tacit practices, fore-sight in terms of our own conceptual framework, and pre-conceptions which tentatively anticipate what is to be discovered provided a philosophic structure encompassing central notions of a tradition in design methodology: Alexander's intuition, Rittel's deliberation from personal viewpoints and Schoen's creation of the design situation.

In Section II, the idea of critiquing systems instead of autonomous expert systems suggested a user-centered approach in which designers could make design moves and be informed of the consequences. Design environments like Janus and Phidias took this idea significantly further, putting the user even more in control, and providing a coordinated system of multiple computational supports for the designing process. But even these systems imposed certain representational constraints that seemed overly restrictive for exploratory domains like lunar habitat design. The shareable, programmable hypermedia of Hermes, described in Section III, allows designers to design their own representational systems for interpreting, framing, and solving problems. By so doing, Hermes tries to support creativity in design, rather than to automate or rigidify the design process.

It may seem that too great a burden is put on the designers by requiring them to construct their own representation schemes and even their own language. Where expert systems had tried to relieve people of the entire problem-solving task, design environments attempted to reach a balance in which domain knowledge was already embedded in the system in ways which could support designers in trying out various design concepts or layouts. The approach of Hermes does in fact adopt this idea of embedding knowledge in the system so that designers do not have to

define all the types, predicates, critics, queries and design rationale from scratch. A "seed" of example definitions and a basic issue-base are provided by the Hermes development team. But, for an exploratory domain like space-based architecture, this seed should be seen as sample building blocks for personal interpretations, rather than representations of accepted knowledge. It is like our proficiency in English and our experience with previous design work: when being creative we must think through new arguments, coin original terms, and sketch innovative images -- but always on the basis of our past insights, vocabulary, and visions.

The Hermes system does not claim to incorporate extensive knowledge of its domain in the sense of an expert system's elaborate set of universally-valid production rules or an expert critiquing system's battery of objective critics. Rather, it provides an environment in which people can view evolving designs from perspectives which are important to them. According to hermeneutic theory, interpretation is more fundamental then knowledge because explicit, propositional knowledge is always the result of interpretive explication. For this reason, the seed in Hermes is conceived of as a form of background history. Just as in normal thought we always interpret based on past experiences and interpretive traditions which we initially accept uncritically, so in Hermes we design using a seed whose every aspect is subject to critique and modification. Over time, this seed evolves, the way human traditions do.

A knowledge-based system like Janus would typically be seeded with information that purports to capture at least parts of an objective theory of the domain. For instance, it might contain an issue-base which contains the primary issues of design in the domain along with the standard options for resolving the issues, a palette of the basic primitive components, and a catalog of prototypical solutions. By contrast, the interpretation-based seed of Hermes provides tools for building interpretive perspectives of domain artifacts; it includes issues, palette items and artifacts that have been constructed under different interpretations in past design projects. Specifically, the Hermes seed consists of information from a series of lunar habitat design sessions that were captured on videotape during preliminary research on the domain, and then modeled in the Hermes system. Additional examples were added from published designs of lunar habitats. Then, an issue-base was constructed to provide a structure to the complex of inter-related rationale issues.

Because lunar habitat design is an exploratory domain, there is no such thing as a comprehensive theory or objective view of the field. Case studies from particular interpretive perspectives provide the only base on which new design efforts can be built. Although it is often possible to systematize the information that has haphazardly accumulated in a seed through use, the result of this kind of "re-seeding" can make no claim to objectivity or comprehensiveness. The act of reorganizing itself proceeds under a certain interpretation or mixture of interpretations of the field, and interesting future designs will focus on new approaches and concerns that were not previously thought of.

Having created a prototype software system that allows designers to fashion their own representational schemes as part of their design work, the next step is to have design teams use Hermes intensively in practice. This will continue the participatory design approach (Ehn, 1988) of computer scientists and domain practitioners working together on the software development. The attempt to use the system will raise new issues and suggest new features to increase the

usability of Hermes. At the same time, it will provide important experience with the evolution of the seed. In particular, it should reveal ideas for making the disclosure language easier to use and for managing the growing network of interpretive contexts.

From a theoretical perspective, Hermes-in-use can provide a laboratory for the investigation of the interpretive process itself. The content of contexts created as designs go through versions can serve as archeological sites documenting efforts at design as interpretation. Beyond the forms of interpretation already supported by Hermes, new forms could be studied, and support for them implemented. Consider three natural forms that interpretation takes in our daily life and work:

* Stream of consciousness narratives we recite to ourselves that integrate and make sense of our goals, actions, interpretations, criticisms, reflections.
* Conversations we have with other people and on-going argumentations or discussions, which provide contexts in terms of which individual words, statements, and images are understood and out of which innovations come to us.
* Expansions of our personal set of beliefs or our worldview, which forms yet another horizon in which interpretation takes place.

The hypermedia structure of Hermes provides a convenient medium for capturing these forms of interpretation and making them available in the process of design. The structure of personal interpretive contexts could be used to keep different people's personal thoughts and beliefs separate, while shared contexts can be used for dialogues, in which many statements are shared but can be interpreted differently and related to different background beliefs. Linked together like a semantic network, argumentation could function computationally as a context for interpreting embedded statements. Operationalizing the hermeneutic dimensions of understanding would create a primitive computer model of interpretation, a contemporary version of the ancient Delphic god. It would broaden the implications of Hermes from a tool for lunar habitat layout to a more encompassing computational medium for supporting interpretation in design.

Acknowledgments

The perspective on design methodology and the approach to computer support for design presented here grew out of the research of Ray McCall of the School of Environmental Design, Gerhard Fischer of the Department of Computer Science, and other members of the Human-Computer Communication group at the University of Colorado at Boulder. The hermeneutic approach stems from the author's participation in Hans-Georg Gadamer's seminars at the University of Heidelberg during 1967/68.

Johnson Engineering (JE) of Boulder contributed generously the time and expertise of Vice President John Ciciora and Designer Mike Pogue. They provided the primary source of information about lunar habitat design, its needs and its methods.

The research in providing computer support for the task of lunar habitat design was supported in part by grants from the Colorado Advanced Software Institute (CASI) for 1990-91, 1991-92 and 1992-93 in collaboration with IBM and JE. CASI is sponsored in part by the Colorado Advanced Technology Institute (CATI), an agency of the State of Colorado. CATI promotes advanced technology education and research at universities in Colorado for the purpose of economic development.

References

Alexander C (1964) *Notes on the Synthesis of Form.* Cambridge: Harvard University Press.

Alexander C (1971) The State of the Art in Design Methods. In Cross N (1984) *Developments in Design Methodology.* New York: Wiley.

Compton WD, Benson CD (1983) *Living and Working in Space: A History of Skylab.* Washington, DC: NASA.

Conklin J, Begeman M (1988) gIBIS: A Hypertext Tool for Exploratory Policy Discussion. *Proceedings of the Conference on Computer Supported Cooperative Work.* New York: ACM. 140.

Coyne R, Snodgrass A (1991) What is the Philosophical Basis of AI in Design? Working paper. Faculty of Architecture, University of Sydney.

Design Edge (1990) *Initial Lunar Habitat Construction Shack.* Design control specification. Houston, TX.

Donald M (1991) *Origins of the Modern Mind: Three Stages in the Evolution of Culture and Cognition.* Cambridge: Harvard University Press.

Dreyfus H (1972) *What Computers Cannot Do.* New York: Harper and Row.

Dreyfus H (1985) Holism and Hermeneutics. In Hollinger R (Ed.) (1985) *Hermeneutics and Praxis.* Notre Dame, IN: University of Notre Dame Press.

Ehn P (1988) *Work-Oriented Design of Computer Artifacts.* Stockholm: Arbetslivscentrum.

Engelbart, D (1963) A Conceptual Framework for the Augmentation of Man's Intellect. In Howerton, P (Ed.) (1963) *Vistas of Information Handling.* (Vol. 1). Washington, DC: Spartan Books. Reprinted in Greif I (Ed.) (1988) Computer-Supported Cooperative Work. San Mateo, CA: Morgan Kaufmann.

Fischer G, Girgensohn A (1990) End-User Modifiability in Design Environments. *Human Factors in COMPUTING Systems, CHI '90 Conference Proceedings (Seattle, WA).* New York: ACM.

Fischer G, Nakakoji K, Ostwald J, Stahl, G, Sumner T (1993) Embedding Critics in Design Environments. *The Knowledge Engineering Review*, Special Issue on Expert Critiquing. Forthcoming.

Fitzgerald F, Rashid R (1986) The Integration of Virtual Memory Management and Interprocess Communication in Accent. *ACM Transactions on Computer Systems*, Vol 4, No 2 (May), 147.

Foley J, van Dam A, Feiner S, Hughes J (1990) *Computer Graphics: Principles and Practice.* Reading, MA: Addison-Wesley.

Freud S (1952) *A General Introduction to Psychoanalysis.* New York: Washington Square Press.

Gadamer H-G (1960) *Wahrheit und Methode*. Tuebingen: Mohr. Translation: Gadamer H-G (1988) *Truth and Method.* New York: Crossroad.

Halasz F (1988) Reflections on Notecards: Seven Issues for the Next Generation of Hypermedia Systems. *Communications of the ACM.* Vol. 31, No. 7.

Hegel GWF (1807) *Phaenomenologie des Geistes.* Translation: Hegel GWF (1967) *Phenomenology of Mind.* New York: Harper & Row.

Heidegger M (1927) *Sein und Zeit.* Tuebingen: Niemeyer. Translation: Heidegger M (1962) *Being and Time.* New York: Harper & Row.

Kant I (1787) *Kritik der reinen Vernunft.* Translation: Kant I (1929) *Critique of Pure Reason.* New York: St. Martin's Press.

Kazmierski M, Spangler D (1992) Lunatechs II: A Kit of Parts for Lunar Habitat Design. Unpublished project report, College of Environmental Design, University of Colorado at Boulder.

Kuhn T (1962) *The Structure of Scientific Revolutions.* Chicago: University of Chicago Press.

Marx K (1967) *Das Kapital.* Hamburg: Meissner. Translation: Marx K (1977) *Capital.* New York: Vintage.

McCall R (1989) Mikroplis: A Hypertext System for Design. *Design Studies*, 10 (4), 228.

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. *Proceedings of the European Conference on Hypertext* (ECHT '90).

Moore GT, Fieber JP, Moths JH, Paruleski KL (1991) Genesis Advanced Lunar Outpost II: A Progress Report. In Blackledge RC Redfield CL Seida SB (Eds.), *Space -- A Call for Action: Proceedings of the Tenth Annual International Space Development Conference.* San Diego, CA: Univelt, 55.

Nilsson N (1980) *Principles of Artificial Intelligence.* Palo ALSO: Morgan Kaufmann.

Norman D (1993) *Things That Make Us Smart.* Reading, MA: Addison-Wesley. In preparation.

Polanyi M (1962) *Personal Knowledge.* London: Routledge & Kegan Paul.

Popper K (1965) *Conjectures and Refutations.* New York: Harper & Row.

Rittel H, Webber M (1972) Dilemmas in a General Theory of Planning. Working Paper No. 194. University of California at Berkeley.

Schoen D (1983) *The Reflective Practitioner.* New York: Basic Books.

Schoen D (1992) Designing as Reflective Conversation with the Materials of a Design Situation. *Knowledge-Based Systems*, 5, 3.

Silverman B (1992) Survey of Expert Critiquing Systems: Practical an Theoretical Frontiers. *Communications of the ACM*, 35, 4, 106.

Simon H (1973) The Structure of Ill-structured Problems. *Artificial Intelligence*, 4, 181.

Simon H (1981) *The Sciences of the Artificial.* Cambridge: MIT Press.

Snodgrass A, Coyne R (1990) Is Designing Hermeneutical? Working paper. Faculty of Architecture, University of Sydney.

Stahl G (1975) Marxian Hermeneutics and Heideggerian Social Theory: Interpreting and Transforming Our World. Unpublished Ph.D. dissertation. Northwestern University.

Stahl G (1991) A Hypermedia Inference Language as an Alternative to Rule-Based Expert Systems. Technical Report CU-CS-557-91. Computer Science Department, University of Colorado at Boulder. Abridged version in *ITS Expert Systems '92.* Forthcoming.

Suchman L (1987) *Plans and Situated Actions: the Problem of Human Machine Communication.* Cambridge: Cambridge University Press.

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

Winston P H (1981) *Artificial Intelligence.* Reading, MA: Addison-Wesley.