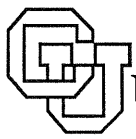
Toward a Theory of Hermeneutic Software Design

Gerry Stahl

CU-CS-589-92 March 1992



University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

Toward a Theory of Hermeneutic Software Design

Gerry Stahl

CU-CS-589-92 March 1992

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

ABSTRACT

This paper proposes dissertation research on a new paradigm of software design based on theories of human interpretation. It is argued that computer supported design environments in certain domains should leave all matters of interpretation to the human user and should try to support the user's interpretation, exploration and development of designs by providing a computationally powerful medium of external memory for storing the evolving design artifact and related knowledge. Hermes, a prototype software environment for the design of lunar habitats, is based on a unified hypermedia knowledge representation system incorporating group and personal perspectives as well as an end-user disclosure language for articulating interpretations of design elements.

Toward a Theory of Hermeneutic Software Design

INTRODUCTION

Prior to the days of professional architects, knowledge of building design could be retained in the heads of individuals, passed down by apprenticeship and tradition, modified by trial and error. In modern times such knowledge has required supports external to the individual mind: increasingly complex drawings; textbooks of examples, rules and data; formal methods of computation. Today, high-tech design projects -- like the development of lunar habitats for astronauts -- call for computer support to help manage the extensive required knowledge.

The challenge for systems to support designers is to facilitate creativity and to enhance control by the users. Hermeneutic software design seeks to achieve this by understanding the process of design with the help of concepts from the philosophy of Martin Heidegger. In particular, four successive stages of human cognition (including design work) are distinguished: *disclosing* the world in which one is situated; being able to *use* things in the world tacitly; *discussing* or reflecting upon or explicitly interpreting something in a particular way; and *analyzing* things theoretically in accordance with formal methods. Each of these stages can be supported by a software system: a *model* of an imagined world can be created with computer graphics; the user can directly *manipulate* representations in this world with a mouse; a simple *language* can be defined for describing represented objects; and a variety of *methods* of computation can be provided for the user.

The process of design can be further conceptualized by extending Heidegger's analysis of the work of art as the opening up of a world in which truth is set to work. This characterization may apply to buildings, at least to great architectural works like the Greek temple in Paestum, but not yet to evolving designs. Rather, design is the tentative, risky process in which one struggles to bring a work into existence. Gradually, the designer creates a new world, explores its possibilities, resolves its conflicts, aligns its constraints, interprets its significances.

A computer system made to support this explorative process can provide an active, multi-media form of external memory for a design team. A graphics component can bring in design elements from catalogs of past designs and palettes of already designed components, as well as presenting the evolving design artifact itself. Textual comments and discussion associated with these graphical objects can communicate design rationale, project requirements, critiquing considerations. Design variations can be maintained for individuals, specific teams or the general public, to facilitate and help structure communication among people involved with a design over a period of time. Formal methods can be provided for computing volumes, costs, topological features, or other concerns. Designers can then use these interlocking tools to vary, test and synthesize elements creatively.

Hermes is a prototype computer system for supporting architectural design. It was devised by studying the process of designing lunar habitats. Using this domain as an illustration, it demonstrates how the variety of forms of knowledge mentioned above can be captured within one representation system, which can then be used to support and integrate a broad range of cognitive functions. The knowledge representation system is a sophisticated form of hypermedia which maintains perspectives and incorporates an end-user interpretation language. This provides the technical basis for promoting designer creativity and control, while facilitating communication among team members.

SCENARIOS

Let us take a look at how Hermes supports the work of design teams.

1. Here is a view of the Hermes interface, with most of its tools represented across the bottom of the window by icons which can be opened into control windows or dialogues. The Graphic Design window shown is a graphics editor, which allows the designer to construct a scale drawing to any level of detail. Typically, components of a drawing will be designed separately or copied from previous designs and modified. Then they will be assembled into a composite drawing. Here, a freezer is being roughed-in for the galley of a lunar habitat. Construction can take place top-down or bottom-up; components can be successively refined or modified; designs are developed through arbitrary iteration. Any object (graphic primitive, design component, top-level design, rationale issue, language predicate, critic, query, etc.) can be named by the user. Any object can have comments and argumentation attached to it. In the Rationale window, the issues which form the basis for the freezer's design rationale are displayed in preparation for review or modification of the discussion. A critic related to freezers is displayed in the Critic Editor window; it can be used to have Hermes critique a design.

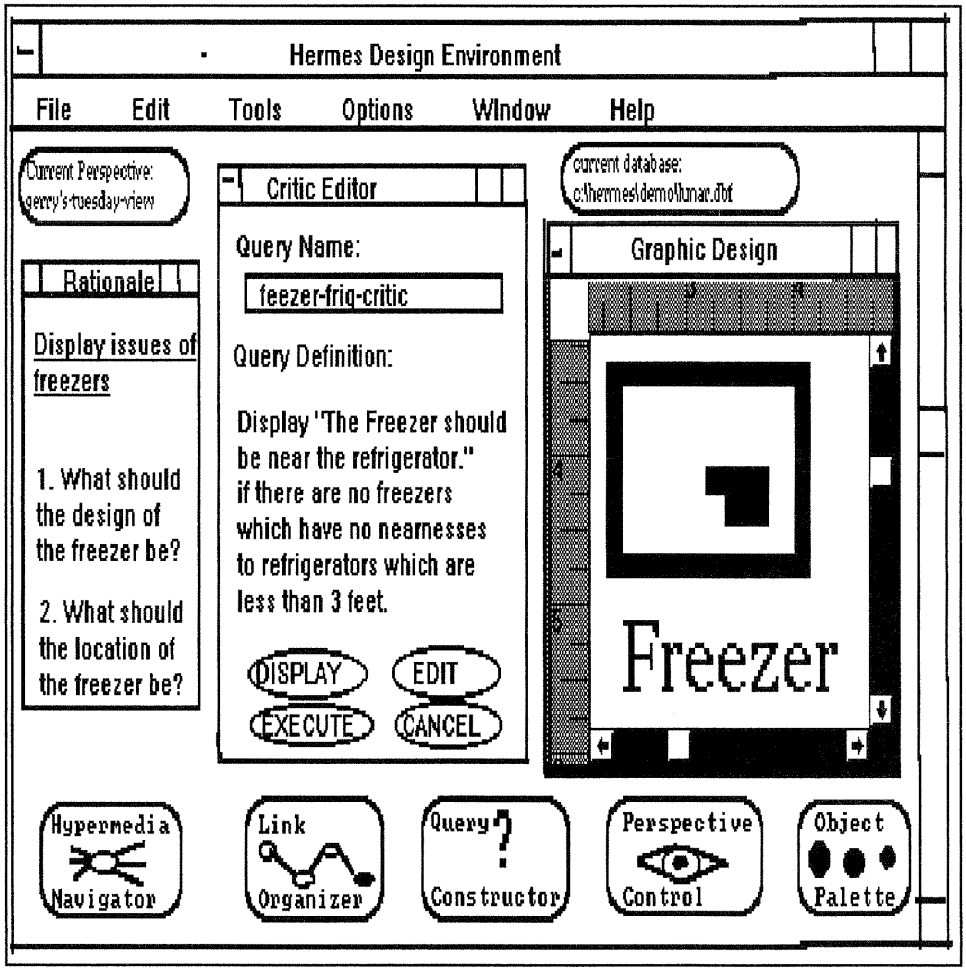


Figure 1. Designing a freezer in Hermes.

2. This is a browser of perspectives of current interest. Three uses of the perspectives mechanism of virtual copies are illustrated here:
(a) Most noticeably there is a hierarchy of public, group and personal perspectives, so the *sharing* of graphic and linguistic designs can be controlled by the users.
(b) Toward the bottom, one sees instances of versioning, so *alternative* design choices can be explored in parallel.
(c) The three beds in the upper right represent a graphical hierarchy in which a single sleep compartment design is viewed three times, with different spatial transformations and other modifications in each instance. The efficiency of *re-use* does not limit user flexibility in customizing each instance.

New perspectives can be created easily and a user can view from within other perspectives (assuming permissions are granted).

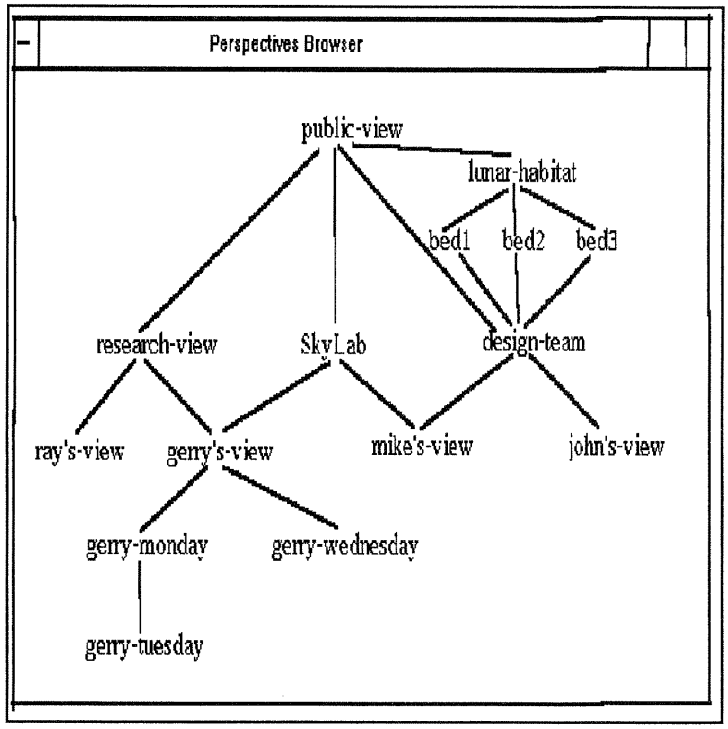


Figure 2. Browser for perspectives.

3. In this scenario, a formal (numeric) computation is defined using the predicate "private-spaces" in a shared perspective. The predicate is in turn modified in two other perspectives. Each perspective defines its own *interpretation* of private space. This means that the result of the computation will be different depending upon the current perspective. Note how the interpretations can be made available for inspection and public discussion by attaching rationale to the definitions.

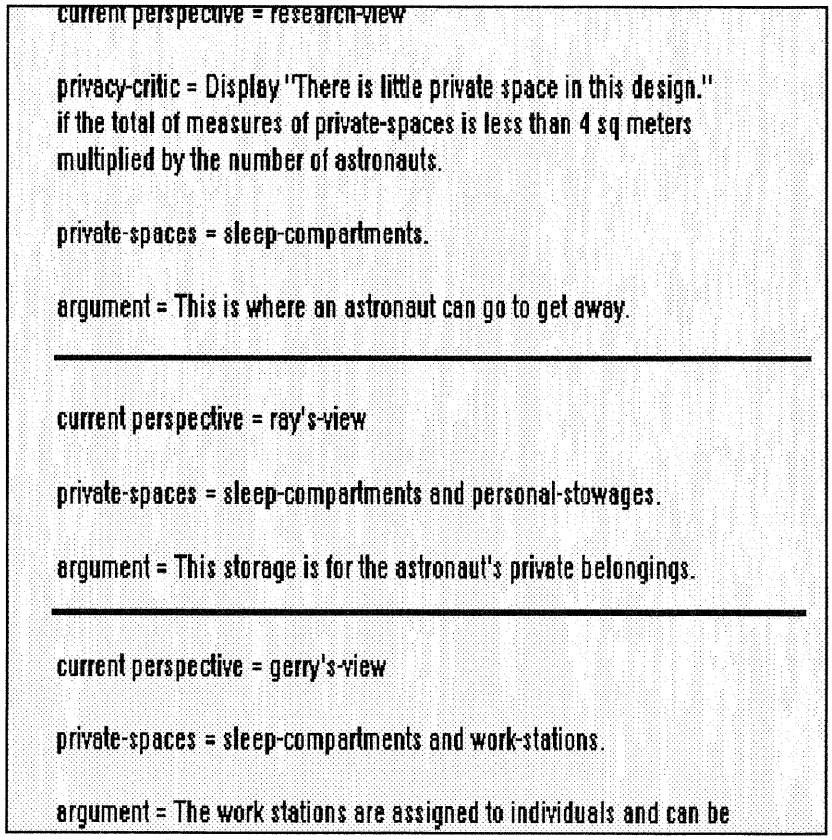


Figure 3. Defining a predicate in 3 perspectives.

4. This query shows how the very general interpretation language can be used to perform system functions, like checking for recent modifications. By default, all objects are time-stamped with hypertext links to creator, date, etc.

This query will produce a multi-media display. The application of a simple query to all forms of knowledge in the system is facilitated by an integrated architecture based on a single knowledge representation structure: hypermedia incorporating the interpretation language and the perspectives mechanism.

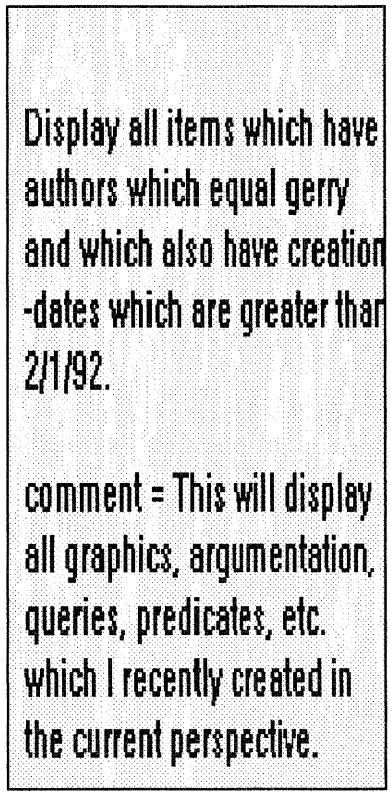


Figure 4. Querying for revisions.

5. The final example shows how the user has quite flexible control over system displays. This is critical to the creativity of the design process, because the displays are the shared external memory. They are the design artifact itself -- "the object to think with" -- that the designers must interpret, explore, synthesize and evolve.

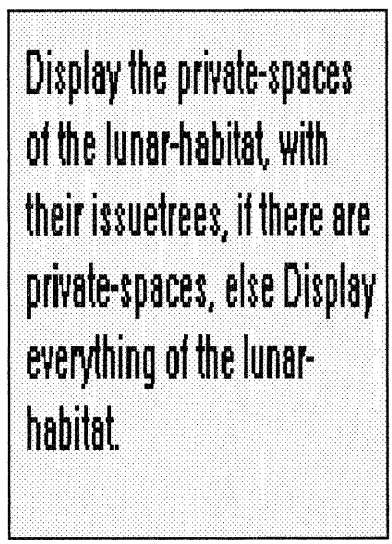


Figure 5. User control over displays.

CENTRAL THESIS

The preceding examples illustrate the approach of hermeneutic software: *The role of the computer system is to support human designers in their task of interpreting, exploring and evolving designs.*

Hermes is based on a cognitive theory of the process of architectural design; it provides an external memory for the design team, including graphic representations and design rationale as defined from multiple perspectives. Integrated with this computationally rich form of memory are a variety of tools to aid manipulation and interpretation, including a special language. This approach incorporates several considerations:

1. Hermeneutic software design defines an open-ended and user-centered approach that contrasts with expert systems based on rationalist models of thought (reliant upon well-defined domain models, explicit computations, and minimal involvement of the user).

2. Design environments can support multiple levels of human cognition: case-based memory, tacit know-how, explicit articulation and methodological calculation (by providing graphic representation, direct manipulation, interpretive language, computation methods).

3. Language is necessary for the development of powerful interpretations, so the user should be given a linguistic facility for control of and interaction with the domain model (especially to name objects, define predicates, formulate views).

4. Shared communication should be supported with a mechanism for distinguishing perspectives (providing personal/group/public versions of knowledge, language, designs).

5. Hermeneutic software can promote designer creativity and user control (with personal perspectives, generative language for interpretation, linguistic control over views). Systems for domain professionals should respect the skills of the users and allow them the option of maximum control, even at the cost of unavoidable conceptual complexities which require learning.

6. An integrated knowledge representation stored in a custom hypermedia system incorporating an interpretation language and a perspectives feature simplifies system architecture (allowing language, perspectives, computational mechanisms to apply to all knowledge, to interrelate it, and to help it evolve) and unifies user interaction (helping the user to feel at home with the system and to synthesize various forms of knowledge represented in it).

THE DOMAIN AND ITS NEEDS

The Hermes system is being developed through a process of participatory design in cooperation with an engineering firm which does design work under contract with NASA, including designs for lunar habitats. Hermes has grown in response to the needs of this kind of work and a study of traditional methods used. The domain of lunar habitat layout is a particularly rich one to investigate and findings in this very specific domain are likely to have broad generality for design, particularly for high-tech architecture.

The need for computer support of lunar habitat design was originally suggested by the sheer volume of knowledge required -- far more than people could maintain in their heads or even locate easily in manuals. In fact, the manuals themselves seem to suffer from a lack of computer-supported maintenance, raising serious questions of how to interpret official regulations consistently with each other. 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. Furthermore, there is a concept of traceability, meaning that there must be documentation tracing how the regulations are incorporated in the design.

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. The design documents are the only form of communication among these many teams. 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 process, with tight cost, weight and volume constraints at every stage, strong rationale must be communicated; a schematic or a pretty picture will not suffice.

Because designers do not have personal experience with life in lunar habitats, knowledge stored in previous related designs (including Skylab, the Shuttle, past trips to the moon) is precious. Old designs are re-used extensively. To the extent that design rationale of the old designs has been captured, it is vitally important. Consequently, it is likely that design rationale will increasingly become an integral part of design. This should add tremendous power to 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 were refined through processes of public critique and evolved into extensive bases of knowledge impossible in non-literate cultures.

There are many other features of the domain of lunar habitat design which influence the functionality of Hermes. For one, it is a form of design in which most components require some amount of customization. One cannot just take a stock sink or bed from a catalog, because of gravitational or volumetric considerations. Even pumps and fans have to be re-thought. So the idea of representing things with schematic rectangles or even with fixed items from a palette is inadequate. 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. Furthermore, there are many design interactions among components that are placed close together -- partially because space is at a premium and because things must work together to form a coherent environment for habitation. This means that design of a given component is very much situated in its context, in terms of neighboring components (e.g., buffering sounds), design concerns (privacy), and projected usage issues (traffic flow). The computer representation of the design must function as the unique world in which situated design can take place effectively.

THEORETICAL CONTEXT

The approach of Hermes -- and particularly its emphasis on language -- grows out of philosophic theory as well as out of design practice. In the most general terms, hermeneutic software design is an attempt to begin to define a human sciences approach to human-computer-communications, in contrast to the natural sciences paradigm of traditional artificial intelligence theory. Philosophically, the twentieth century has witnessed the debate between two general camps. The first is the rationalism which derives from Plato, Descartes and the neo-Kantians, takes credit for the successes of the physical sciences, and is to blame for the excesses of positivism and behaviorism. The other is a tradition that insists that human sciences (like anthropology) require human understanding and the interpretation of subjective meanings, rather than just objective explanations. The rationalist position seems to be fighting an increasingly futile defensive battle in recent years, having received its most thorough-going (though little-understood) philosophical critique from Heidegger. Within computer science, this can perhaps best be seen in the increasing number of influential statements of situated cognition.

Schoen (1983), for instance, contrasts the situated seeing-as and doing-as of knowing-in-action with the explicit, theoretical propositions of technical rationality. For him, the designer constructs and manipulates virtual worlds; in making sense of a unique situation, the designer sees it as something else which is familiar and places it within familiar, named categories. This is a theory of interpretation, as opposed to a rationalist theory of propositional goals. Unfortunately, Schoen does not draw the implications of this theory for programming.

Dreyfus (1991) does argue directly against AI as a model of human cognition, providing an influential commentary on Heidegger (1927). But there are several shortcomings of this attempt: Dreyfus' readings of being as social practices and of breakdowns as the mechanism causing explication are idiosyncratic and limited; his argument does not include Heidegger's later thought; he proposes no alternative to the AI he dismisses.

Coyne (1991) explicitly views design as a process of interpretation and calls for hermeneutic programming based on Heidegger and Gadamer. But he too misses the power of Heidegger's later work, and so his notion of "available architecture" is simplistic, ignoring how designs resolve constraints and how buildings work to define contexts for dwelling. His ideas for software point in the direction of Hermes, but are utterly vague.

Suchman (1987) emphasizes the contrast between rationalist plans and situated action. She also stresses the role of language in constituting human interpretation of situations. But she is concerned with human-computer-communication in which the computer must understand and act (produce copies), rather than with the computer as external memory or as a medium for shared human cognition. Rather than proposing an alternative to rationalist programming, she fine-tunes traditional programs with more sensors of the human situation and with more situated testing of the communications.

Another clear critique of AI from a Heideggerian perspective is presented by Winograd and Flores (1986), who do call for a new approach to software design. They note that the computer is ultimately a structured dynamic communication medium and they stress the central role of language in coordinated action. They propose the Coordinator program as an example of new software as a medium for CSCW and note its limitation: "In many contexts this kind of explicitness is not called for, and may even be detrimental" because language is ultimately an "open-ended domain of interpretation." Despite this recognition, they propose software which failed to be accepted in many social settings because it imposed a rigid, explicit, public structure where people often want to remain implicit, and so it did not empower personal interpretations.

Participatory design, as described by Ehn (1988), is a method for developing software in partnership with the end-users, so it can be designed to support skillful work and democratic workplace relations, in contrast to traditional automation approaches. The idea is to design computer tools for experts which support and extend their skills, including their tacit know-how. As an example, the Utopia project pioneered a desktop publishing toolkit for graphic layout professionals, rather than the automated systems that were putting these experts out of work. This toolkit approach may have been innovative at the time, but is now standard.

HERMENEUTIC THEORY

Hermeneutic software design revisits the philosophy of Heidegger -- which underlies the theories of situated cognition -- to develop a fuller foundation for a new paradigm of programming. It starts from the position that Searle (1980) opposed to "strong AI", namely that only humans (and not computers) have intentionality, that is the ability to *interpret* semantics. Given this in-principle limitation of computers, they need to concentrate on providing computationally-active extended memory for people. Clearly, computers can be most useful in doing this if they present their stored knowledge in a format appropriate to human cognition. As Norman (1993) says, "Without someone to interpret them, cognitive artifacts have no function. That means that if they are to work properly, they must be designed with consideration of the workings of human cognition."

Human cognition is a complicated business. A recent analysis of its structure by Donald (1991) based on anthropological, neurological and linguistic evidence suggests four stages in its historical development (all of which remain still active): *episodic* (case-based), *mimetic* (tacit, gestural), *mythic* (linguistic) and *modern* (based on extended memory: pictures, writing, computers). Heidegger's (1927) philosophical analysis of the logical structure of human being-in-the-world can be comprehended as a parallel to this sequence: *understanding* (the world is disclosed to us as meaningful), *interpretation* (we can make things stand out *as* what they are), *assertion* (we can name and talk about things), and *theory* (we can state propositions using a formal method). Hermeneutic software design aims to support each of these modes of human cognition.

Heidegger goes on to discuss how we understand our situation, the things disclosed within it, and ourselves. This part of his philosophy is called "hermeneutics", which builds on a tradition in the theory of interpretive human sciences and which was subsequently expounded by Gadamer (1960). Understanding is basically a synthesis which brings together our *background knowledge* (previous experience), a *preconception* (expectation, anticipation, tentative conceptual framework), and *foresight* (perspective, point of view). For Heidegger the synthesis is a temporal process that unifies our past (background), present (perspective) and future (projections). We always find ourselves already in a situation -- namely the result of our past interpretive activity. This can perhaps best be illustrated using Schoen's characterization of design: the designer draws upon past experience, tries interpretive moves, and looks for consequences from the perspective of a set of concerns. The world is disclosed anew to us based on our interpretive synthesis; things are discovered in the world as what they are in the new interpretation; the world (in Schoen's phrase) "talks back" to the designer.

The notion that the world talks to us is reminiscent of Heidegger's later philosophy. In discussing art works, Heidegger (1950) talks about the work opening up a world in which truth or being is presented. If one extrapolates from his theory of finished art works (including architectural works), one can propose an ontology of designs as emergent works. In an evolving design, a world is partially, tentatively, progressively disclosed. This world does not have the unity, consistency, simplicity, beauty of a work of art. A bold stroke of the designer's pencil (or mouse) in a concept sketch may declare a new distinction, define a limit, juxtapose conflicting tendencies, reorganize relationships, catalyze the dialectic of part and gestalt. Each stroke threatens to either coalesce or shatter the unity of the nascent concept. If Hermes is to succeed according to this theory, it must act as the medium in which worlds can be set to work, explored and evaluated.

COMPARISON WITH JANUS AND PHIDIAS

Hermes builds on research software models developed at the University of Colorado. It attempts to reinterpret and extend the Janus and Phidias systems within the paradigm of hermeneutic software design. Janus and Phidias are design environments that have both construction-kit palettes of graphic representations and issue-base textual argumentation. Janus provides a catalog of design cases and a critic mechanism. Phidias provides for authoring of issues, primitive graphics editing, and issue-base querying tools.

Both systems suffer from a lack of integration of text and graphics (no unified knowledge representation) and from program design decisions which limit their ability to evolve into real-world systems. In particular, Janus provides little user control over graphics and none over the issue-base. Its computational power (e.g., inferencing capability, calculation of distances) is virtually inaccessible to the user. Phidias' graphics cannot easily be restructured once defined because of their reliance on the PHIGS hierarchy approach. Graphics and issues are only linked at certain fixed points in Phidias. There is no critic mechanism. The query language is limited: it does not support naming, interpretation, perspectives or queries over graphics. Generally, Janus and Phidias are still conceived on the expert system model, where the software works mainly behind the scenes, and the computational power is not available to the user directly.

For instance, in Janus to add a freezer to a kitchen design (assuming the class of freezers has not been defined) requires following an explicit plan (vs. engaging in situated action). First define the freezer class: choose menu item "new object"; choose menu item "design unit class"; assign to class: attributes, descriptions, superclasses, and other info used internally by system; assign to class every critic rule that should apply. The user may need to step back several levels to achieve subgoals: defining new critic rules, relations, global descriptions, inference rules, etc. Each subgoal involves understanding system concerns, menus, property sheets, internal workings and Lisp syntax. None of this is analogous to working in the domain. The plan followed must be one based on an understanding of how the program works internally -- user testing showed that users are forced to follow the plan conceived by the programmer. Furthermore, all the power of naming and inference which are available through Lisp are hidden from the user. The primitive uses of these capabilities which are passed on to the user require the navigation of many levels of menus. Symptomatically, although all of Janus' critics rely upon distance measures, there is no sense of scale apparent to the user in the graphic workspace.

As illustrated in the first scenario above, to add a freezer to a galley design in Hermes involves designing the freezer graphically and attaching argumentation, including critic statements formulated in the Hermes disclosure language. The freezer can then be named and saved as a palette item if desired. Rather than designing from scratch, a virtual copy could be made of a refrigerator with its attached argumentation, and then modified. While this may or may not be easier in Hermes than in Janus for users of various backgrounds, at least in Hermes the user remains within the graphics and language of the domain world of design. The Hermes interpretation language and perspectives mechanism require no knowledge about the software implementation; once the user knows how to use them, all the power of Hermes is available for the user to modify, interpret, query, critique or display any combination of objects in the system.

HERMES LAYERED ARCHITECTURE

To provide full user extensibility, all text, graphics and language constructs are stored as data. So Hermes is fundamentally an object-oriented data-base system (optimized to be efficient and to scale up). All knowledge is represented in a custom hypermedia system which incorporates the interpretation language in nodes and the perspectives mechanism in links.

The perspectives mechanism is based on a virtual copying scheme like that in the NeXT operating system, which is designed to conserve computer memory. In Hermes this mechanism is re-worked to support human communication and shared understanding in cooperative work contexts.

Similarly, the interpretation language is based on a query language, inference language or end-user programming language approach. In Hermes it is developed into a tool for viewing data through an interpretation. Significantly, statements in the language (like previously defined critics, queries, predicate definitions, conditional clauses, and display criteria) are readily interpretable by the user. All hypermedia navigation in Hermes is done by means of the language and perspectives, so that all operations and displays can be controlled (interpreted) by the user. The language syntax is quite expressive and capable of arbitrary complexity (nesting, macro expansion, recursion).

The Hermes database is built up in layers:

(a) During creation of a new Hermes system, a number of data items (link types, predicates requiring special programming) are created.

(b) Then the database is "seeded" by a team of domain experts and knowledge engineers. The seed includes media primitives for the domain (e.g., graphic attributes; sound volume, timbre, duration; video forward), useful predicates, standard critics, sample designs, common components, and a structured issue base for the domain. The seed might include supplementary hypertexts of argumentation, advice, regulations and information (such as ergonomic data or rules of thumb).

(c) Through use, the database gains a rich history of data related to new design efforts.

(d) Finally, a current project is developed which can draw upon knowledge stored at any of these layers.

The Hermes interface conforms to the MS-Windows standard. It is designed to shelter the naive user from the potential complexity and power of the system's user control. Methods of re-use and iterative construction are emphasized. The seed provides the language constructs most likely to be needed. Because they can be easily read and intuitively understood, these constructs can be readily explored, re-used and modified. The interface and the language also encourage modular and iterative building up of complex structures. Browsers provide quick access to previously defined objects. Hermes aims to support creative design by permitting arbitrarily innovative constructions of graphics and argumentation, while providing an intuitive and uniform interface which will not distract the user from domain concerns.

WORK PLAN

Tasks Done:

The goal of developing a new programming paradigm like hermeneutic software design is a "wicked problem" indeed. The general outlines of the project have now been defined and the remaining work is largely a matter of fleshing them out. The core philosophic issues have been identified and an initial review of the most important texts has suggested the tentative conceptual framework.

Close collaboration with experts in the domain of lunar habitat design has produced an initial understanding of their pressing needs and their traditional methods. About thirty hours of videotaped design sessions provides ample material for further study.

The primary software modules for Hermes have already been developed: the database system, hypermedia, perspectives, a preliminary version of the language, and textual interfaces for testing.

Tasks To Do:

The hermeneutic philosophy and ontology of design must be developed through close textual reading and consideration of the design domain and the software prototype. This philosophy needs to be explicated and presented for a lay audience.

The videos have to be analyzed in detail to produce an inventory of what designers do and what supports or tools they need. Specifically, how do they use measurements, language, graphics? Any conclusions along these lines should be checked with the domain experts.

Before the code for the interpretation language is rewritten, a program walk-through of the language will be performed. The language will be revised according to the results of this and then implemented. Much work must be done on the windowing interface. While some initial concepts have been tried, it is likely that many iterations of design and testing will be necessary to achieve the required naturalness of feel, even for demo purposes.

Demos for the domain experts will start as soon as an initial version of the system is ready. This will require the definition and input of the seed data: issue-base, sample designs, predicates, critics, etc. The seed data will be derived from the videos. Full-blown scenarios in the domain will be defined and carried out. This will be an important learning experience about the limitations of Hermes and needed revisions. The scenarios will be carefully reviewed by the domain experts. They may be taped and circulated within NASA and Houston as well.

Finally, the dissertation will be written, explaining the project, expounding the theory, reviewing the participatory design process and describing the prototype software.

Post-Doc Tasks:

Part of the concept of Hermes is that it could become a real-world system, used by a team of environmental design students doing a lunar habitat project or even by a NASA design group. This would require considerable refinement of the software and the addition of more functionality. It would be nice to implement 3-D graphics, adjacency constraints and an inheritance mechanism, for instance. A cognitive walk-through, user testing and more demos would drive further interface development. Real-world testing would be an important step to take, and one which would provide important insights into Hermes as a work-oriented tool or a medium of shared communication in a real situation. However, this is beyond the scope of the dissertation.

BIBLIOGRAPHY

Adorno, T.W. (1964). *Jargon der Eigentlichkeit: Zur deutschen Ideologie* [The Jargon of Authenticity]. Frankfurt am Main: Suhrkamp.

Adorno, T.W. (1966). *Negative Dialektik* [Negative Dialectics]. Frankfurt am Main: Suhrkamp.

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

Alexander, C., Ishikawa, S., Silverstein, M. (1977). *A Pattern Language*. New York: Oxford University Press.

Bell, B., Citrin, W., Lewis, C., Rieman, J., Weaver, R., Wilde, N., Zorn, B. (1992). *The Programming Walkthrough: A Structured Method for Assessing the Writability of Programming Languages*. Technical Report CU-CS-577-92 January 1992. University of Colorado.

Bush, V. (1945, June). As We May Think. *Atlantic Monthly*, 176 (1), 101-108. Reprinted in Greif (1988).

Carroll, J.M. & Kellogg, W.A. (1989). Artifact as theory- nexus: hermeneutics meets theory-based design, *Proceedings of the Conference of Human Factors in Computing Systems*, Austin, 7-14.

Coyne, R. (1991). Inconspicuous Architecture, *Gadamer Action & Reason: Conference Proceedings*. Australia: University of Sydney.

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

Dreyfus, H. (1965). *Alchemy and Artificial Intelligence*. The RAND Corporation.

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

Dreyfus, H., ed., (1982). *Husserl, Intentionality, and Cognitive Science*. Cambridge: MIT Press.

Dreyfus, H. & Dreyfus, S. (1986). *Mind Over Machine*. New York: The Free Press.

Dreyfus, H. (1991). *Being-in-the-World: A Commentary on Heidegger's Being and Time, Division I*. Cambridge: MIT Press.

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

Eisenberg, M. (1991). Programmable Applications: Interpreter Meets Interface.

Engelbart, D. (1963). A Conceptual Framework for the Augmentation of Man's Intellect. In P. Howerton (Ed.). (1963). *Vistas of Information Handling* (Vol. 1). Washington, DC: Spartan Books. Reprinted in Greif (1988).

Ericsson, K.A. & Simon, H.A. (1984). *Protocol analysis: verbal reports as data*. Cambridge: MIT Press.

Fischer, G., McCall, R., & Morch, A. (1989, November). Janus: Integrating Hypertext with a Knowledge-based Design Environment, *Proc. of Hypertext '89*, Pittsburgh, PA: ACM, 105-117.

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., Grudin, J., Lemke, A., McCall, R., Ostwald, J., Reeves, B., Shipman, F. (1991). Supporting Indirect, Collaborative Design with Integrated Knowledge-Based Design Environments. Submitted to *Human- Computer Interaction*.

Gadamer, H. G. (1960). *Wahrheit und Methode* [Truth and Method]. Tuebingen: Mohr.

Gadamer, H. G. (1966). Die Universalitaet des hermeneutischen Problems [The universality of the hermeneutic problem]. In H. G. Gadamer (1967). *Kleine Schriften I Philosophie Hermeneutic*. Tuebingen: Mohr.

Greenbaum, J. & Kyng, M. (1991). *Design at Work: Cooperative Design of Computer Systems*. Hillsdale, NJ: Lawrence Erlbaum.

Greif, I. (Ed.) (1988). *Computer-Supported Cooperative Work*. San Mateo, CA: Morgan Kaufmann.

Habermas, J. (1968). *Erkenntnis und Interesse* [Knowledge and human interests]. Frankfurt a. M.: Suhrkamp Verlag.

Habermas, J. (1985). *Der philosophische Diskurs der Moderne: Zwoelf Vorlesungen* [The Philosophical Discourse of Modernity]. Frankfurt am Main: Suhrkamp.

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

Heidegger, M. (1927). *Sein und Zeit* [Being and time]. Tuebingen: Niemeyer.

Heidegger, M. (1971). *Poetry, Language, Thought*. Trans A. Hoftadter. New York: Harper & Row.

Heidegger, M. (1975). *Der Grundprobleme der Phaenomenologie* [Basic problems of phenomenology]. Gesamtausgabe vol. 24. Frankfurt am Main: Klostermann.

Heidegger, M. (1979). *Prolegomena zur Geschichte des Zeitbegriffs* [Introduction to the history of the concept of time]. Gesamtausgabe vol. 20. Frankfurt am Main: Klostermann.

Heidegger, M. (1950). Ursprung des Kunstwerks [The origin of the work of art]. In M. Heidegger (1950). *Holzwege*. Frankfurt am Main: Klostermann.

Heidegger, M. (1953). Wissenschaft und Besinnung [Science and reflection]. In M. Heidegger (1954). *Vortraege und Aufsaetze*. Pfullingen: Neske.

Hegel, G. W. F. (1833). *Grundlinien der Philosophie des Rechts* [Principles of the philosophy of right]. Leipzig.

Illich, I. (1973). *Tools for Conviviality*. New York: Harper & Row.

Kunz, W. & Rittel, H.W.J. (1970). *Issues as Elements of Information Systems*. Working paper 131. Center for Planning and Development Research, University of California, Berkeley.

Lakoff, G. (1987). *Women, Fire, and Dangerous Things*. Chicago: Univ. of Chicago Press.

Lefebvre, H. (1991). *The Production of Space*. Oxford: Blackwell.

McCall, R. (1987). PHIBIS: Procedurally Hierarchical Issue- Based Information Systems. *Proceedings of the Conference on Architecture at the International Congress on Planning and Design Theory*. New York: American Society of Mechanical Engineers.

McCall, R. (1989). Mikroplis: A Hypertext System for Design. *Design Studies*, 10 (4), 228-238.

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)*.

Minsky, M. (1985). *The Society of Mind*. New York: Simon and Schuster.

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

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

Norman, D. & Draper, S. (1986). *User Centered System Design: New Perspectives on Human-Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum.

Palmer, R. (1969). *Hermeneutics: Interpretation Theory in Schliermacher, Dilthey, Heidegger and Gadamer*. Evanston: Northwestern University Press.

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

Putnam, H. (1967). The Nature of Mental States. In N. Block (Ed.). (1980). *Readings in Philosophy of Psychology* (Vol. 1). Cambridge: Harvard University Press. First published as Psychological Predicates. In W. H. Capitan & D. D. Merrill (Eds.). (1967). *Art, Mind and Religion*. Pittsburgh: University of Pittsburgh Press.

Putnam, H. (1988). *Representation and Reality*. Cambridge: MIT Press.

Richardson, J. (1991). *Existential Epistemology: A Heideggerian Critique of the Cartesian Project*. Oxford: Claredon Paperbacks.

Rittel, H. & Webber, M. (1973). Dilemmas in a general theory of planning. *Policy Studies*, 4, 155-69.

Rorty, R. (1977). *Philosophy and the Mirror of Nature*. Princeton: Princeton University Press.

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

Schutz, A. (1970). *Reflections on the Problem of Relevance*. New Haven: Yale University Press.

Searle, J. (1980). Minds, Brains, and Programs. *The Behavioral and Brain Sciences*, vol. 3.

Searle, J. (1983). *Intentionality: An Essay in the Philosophy of Mind*. Cambridge: Cambridge University Press.

Simon, H. (1973). The Structure of Ill-structured Problems. *Artificial Intelligence* 4, 181-200.

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, Spring). *Marxian Hermeneutics and Heideggerian Social Theory: Interpreting and Transforming Our World*. Ph.D. Dissertation, Northwestern University.

Stahl, G. (1975, Winter). *The Jargon of Authenticity: An Introduction to a Marxist Critique of Heidegger*. Boundary 2, vol. III, no 2.

Stahl, G. (1976, Winter). Attuned to Being: Heideggerian Music in Technological Society. *Boundary 2*, vol. IV, no 2.

Stahl, G. (1989, February). *Philosophy of Metaphor:Implications of Philosophy for AI*. Presentation for Issues in Lexical Semantics course.

Stahl, G. (1989, March). *Arguments for Equality: Social Implications of the Euclid Argumentation Software*. Paper for Advanced AI Programming course.

Stahl, G. (1989, April). *Euclidean Reasoning versus Gentle Persuasion: a Matter of Semantics*. Paper for Issues in Lexical Semantics course.

Stahl, G. (1989, November). *The Proper Treatment of Representation*. Paper for Foundations of Cognitive Science course.

Stahl, G. (1991, January). *The Philosophical Context of AI and the Situated Cognition Paradigm Shift*. Presentation for Design Seminar.

Stahl, G. (1991, February). *The Relevance of Artificial Intelligence: What Relevant-knowledge-based Systems Could Do*. Paper for Design Seminar.

Stahl, G. (1991, March). *PhiQL: Formal Specification of a Query Language for HyperMedia*. Unpublished technical report.

Stahl, G. (1991, April). *Beyond Rational Design*. Presentation for Design Seminar.

Stahl, G. (1991, October). *Intelligent Hypertext as an Alternative to Expert Systems*. Presentation at Colorado Institute for Artificial Intelligence Research Symposium, Denver.

Stahl, G. (1992). *A Hypermedia Inference Language as an Alternative to Rule-Based Expert Systems*. Technical Report CU-CS-557-91. Computer Science Dept., University of Colorado a tBoulder.

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

Suchman, L. & Trigg, R. (1991). Understanding Practice: Video as a Medium for Reflection and Design. In Greenbaum & Kyng (1991).

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.

Wixon. D., Holzblatt, K. & Knox, S. (1990, April). Contextual Design: An Emergent View of System Design. *CHI '90 Proceedings.*