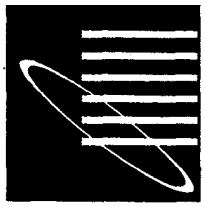

Putting Interactive CAD Systems in Perspective

Paul Taplin
The University of Adelaide

An overview of the principles used to develop productive interfaces is presented, and afresh approach to the design and use of Simulation Systems is suggested. 7he notion of "perspective" (a way of viewing the HumanComputer Interaction) is introduced and then extended with analogy. The aim is to encourage a complete exploration of the roles that users play when interacting with computer systems and to ensure that adequate user models and functionality are developed

Introduction

"The problem of the user interface might reasonably be described as the last frontier in computing. It is not a problem that can be solved by using faster, more powerful computers. It is a problem whose solution depends primarily on our understanding of how people think and work. It is a design problem rather than a computing problem". (Hicks and Essinger 1991, p.50)

Computer technology has permeated nearly every facet of architectural practice where traditional tools have previously been used; we are seeing computers taking on roles and supporting us in management, information processing, solution generation and solution evaluation. It is in the area of solution evaluation that we find simulation tools rapidly increasing in power, with models that are becoming more complex and computers that are becoming faster. Many software designers (not only in this field) are concerning themselves only with improving efficiency and performance of the hardware and software, and are too busy with these technical problems to deal with the human issues. This one-sided approach fails to improve the efficiency and performance of the human-computer system, chiefly because the computer is only perceived as being a glorified tool, and not a partner with the human to undertake a task.

The purpose of this paper is not to describe or promote a particular building performance system, but rather to encourage the re-thinking of the design of the interface to such systems by making their interfaces accessible to architects and other building professionals. Its premise is that the technical capabilities of existing systems are not being explored in practice because they are simply too difficult to use.

My current research concerns the investigation of ways of establishing productive interfaces between designers and systems for generating designs. These systems couple the characteristics and strengths of computational systems with the cognitive capabilities of the designer. The cooperative nature of these (relatively) novel systems led to an understanding that the way in which the user is to relate to the computer requires thorough examination, and that a creative approach, with fresh ideas, is essential. Hicks and Essinger express this need well in the quotation that heads this section.

The importance of the interface

Interactive computer systems *are* difficult to design. The single, command-line interfaces of systems have been replaced - almost exclusively - by graphically and spatially oriented interfaces. Such systems can now represent real or imaginary worlds, and corresponding functions, through the use of metaphor. These interfaces offer visual and cognitive representations to the user who, as a result, does not have to work so hard (cognitively) to produce a mental model of the system. Many systems (simulation systems among them) are based on older, more familiar systems that evolved from command-line-driven interfaces. In "renewing" such interfaces, an attempt is often made to adopt user interface standards and incorporate features that might be called "user-friendly" (for example, being mouse and menu-driven) but these alone will not necessarily make a system more useable. There is a

Author's address: Department of Architecture, University of Adelaide, S.A. 5005, Australia.
Tel. + 61 8 303 5836, Fax. + 61 8 303 4377
E-mail: *ptaplin@arch.adelaide.edu.au*

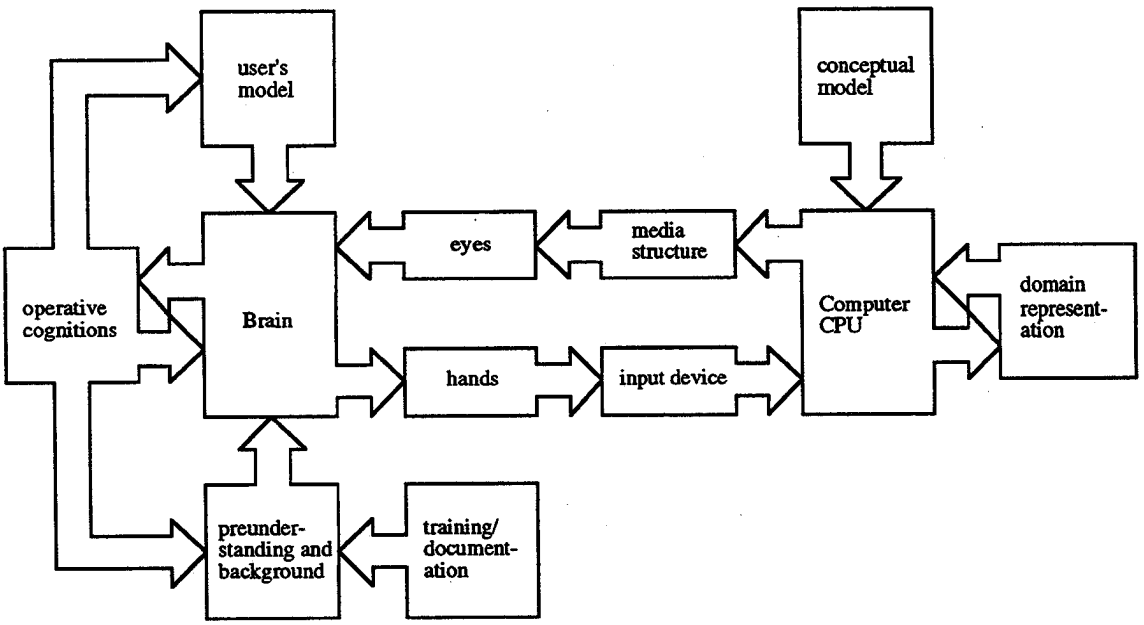

FIGURE 1: Model of the Interaction Process

failure to understand that in writing software we are inviting communication between the user and the computer; not only that but, "The design of computer systems [is] the design of a medium for communicating with other humans." (Cox and Walker 1990, p.ix). That is, the user is in communication with the system designer through the interface that the designer builds.

The interface treads a fine line between showing the user what the computer is doing (because it works invisibly) and shielding them from the complex workings which they do not need to see; people do not use algorithms or data structures when they use computers, they make selections, give commands, and manipulate features of the interface. First and foremost, the interface must be designed with the needs of the user in mind.

We can view the computer as a *partner* who may possess and help us *use* tools instead of demoting and under-utilising the potential of it. In understanding this vision, Licklider proposed a "man-machine symbiosis" (Licklider 1960), in which a synergistic coupling of human and machine capabilities would occur. This tight coupling, where both partners would be dependant on the other, could not be implemented with the technology of the day because systems were not interactive and did not provide feedback quickly enough. This situation does not present itself today, and *now* would seem to be the time to continue this research idea again.

People using computers relate what they experience during use with what they know, using the processes of metaphor, analogy, and modelling. These processes (as will be shown later) also have the important roles of idea generation and exploration, for the design of interactive systems. As these processes will be referred to often, a description of them in relation to computer systems will be given so as to avoid confusion.*

Metaphor, a word often connected with current computer systems, represents the computer with objects or events from another domain. (The most common and popular would be the "desktop" metaphor.) *Analogy* is a comparison between objects or events that have different representations but serve the same purpose. The use of analogy involves the recognition of a relationship between objects and the mapping of it to another domain (typically seen when a user is introduced to new computer equipment or environments). A *model* is a representation of relevant properties of any object, mapped into a particular medium (for example, the brain, computer code, or paper); typically they are seen as a representation of the abstract, conceptual structure of a computer system (an example being the representation of a system directory as a tree structure).

As a focus for this discussion on the importance of the different ways of viewing a cooperative computer system, a model of the interaction process is presented (FIG 1).** The user interface forms an information channel, allowing the use of the computer's facilities (without which it would be useless), and consists of the following components:

* Wozny (1989) presents an excellent outline of the differences between metaphor, analogy, and models, and discusses their importance to system design.

** The diagram presented is my own, built upon the terms used in Kammersgaard (1990), and borrowed from Winograd and Flores (1985). It highlights the similarities and symmetries between the two participants. Any misrepresentation is purely my own misunderstanding.

• the *user's model*, which is an idea in the user's mind of what the system does and how it is performed, influencing the way that the user interacts with and accumulates knowledge about the system.
• the *command language* refers to the computer input (for example, keystrokes and gestures with a mouse).
• the *information display* (most often a video monitor) displays the represented task domain.
• *feedback*, which is the computer output as a reply to the commands given by the user. (Feedback might just involve seeing a cursor being moved across the screen.)

All four components react together to maintain a cycle of communication or dialogue.
Expanding these components into concepts that will be used later, are:

• *operative cognitions* that change during the interaction process (that is, the user's "working knowledge" of the system and the task being performed.)
• the *domain representation* which is the computer's representation of the task performed.
• the *media structure* which is a visual (or auditory) representation of the domain representation. It is modified whenever the underlying domain representation is changed, or the user interacts with the computer.
• the user's *preunderstanding and background* which is formed before any interaction (by training or instruction) and also as a consequence of the user's thinking and actions. It acts to limit or direct the user's way of thinking.

Reiterating an important concept, the user model is an image of what the user is doing; a conglomeration of their expectations, preconceptions, existing knowledge and previous experience and will be based upon the things they learn or perceive about a computer system.

Approaches to interface construction

In the 1960's and 70's, users and workplaces were typically made to conform to the needs of the computer systems; humans were required to "think" like machines. The developers and operators of these systems needed to be experts at their work and acted as interfaces to the people who needed to make use of the results obtained by the machines. With the increase in power, and decrease in size and cost of computers, the end-user also became the operator. It was during the 1980's that an inversion of design took place, in which hardware and software needed to be fitted to the requirements of the user and their ways of thinking

Historically, the approach to software design was to make use of "Fitt's List" (Singleton 1974) in which the objectives and functions of the application were defined and the tasks allocated to the human or computer on the basis of ability. (In combining the best attributes of both it was hoped, and seems reasonable, to achieve the best overall effect, in which the two combined are more powerful than each separately.) The importance of task analysis was acknowledged by the Xerox Star design team who commented that "Task Analysis is the first part of interface design"(Smith et. al. 1982).

One can begin development of an interface knowing the functionality required, and then building an interface in front of it, or by first beginning with an interface and building the functionality behind it (making the implementation more concrete). This latter method was followed by the Xerox Star design team, whose metaphor of a desktop/office environment - which led to the development of the Apple Macintosh interface - was designed before the full functionality of the system was developed. (Xerox spent several work-years at the outset of the project to discuss and develop a model for their office information system.) In starting with the interface, and using the conceptual model of a desktop/office it was found that the approach changed or determined the functionality of the system. (For example, most systems of the time that supported the exchanging of files and electronic mail did so with separate systems. In the physical offices around, it was observed that the distinction between the two objects was not made. As a result, the functionality and interface were modified so that consistency was maintained with the real-life example.) Using the right metaphor, consistency was maintained, and this in turn led to simplicity.

The importance of mental models can be seen when investigation is made into the differences between expert and novice users of a system. Experts have a detailed, abstract knowledge, and use analogy to relate old domains to new domains (for example, when using new computer hardware or software). Novices, on the other hand, initially learn a system through metaphor and analogical processes; as their knowledge increases, a model of the system develops. The mental model is later used to interact with and predict the behaviour of the system - similarities and differences will be noted, and either added to or removed from the model.

Infrequent users rarely ever relate their knowledge of the system back to their mental model. Each time they return to the system, they bring with them their old model, so that if a situation occurs that does not fit the behaviour they imagined, confusion occurs. As Wozny notes, "...the world of the novice or infrequent user is characterised by confusion and discomfort in the absence of any clear model or direction for thinking about a system" (Wozny 1990, p.279).

Users create models regardless of whether a source model exists in a system or not. (Humans by nature are mental model builders - a cognitive function of trying to order their perceptions and experiences.)

For the designer, the user's model presents an abstraction of the user's requirements and the way

in which they might react with the system. A system (or any tool) that behaves as the user expects will be regarded as a "good" tool, characterised by:

• *User Control*, where the user is in charge at all times and understands or sees the sequences of operations performed by the computer (in necessary situations).
• *Transparency*, where the user sees their operation on the domain representation without being focused on the issues of interacting with the interface. (This leads to a reduced cognitive load.)
• *Learnability*, where the user is able to perform basic tasks with relative ease, and an increase of skills is supported.

These desirable characteristics are direct results of the close mapping between the user's model and the conceptual model (presented in the interface and documentation).

The designer has two options to achieve this close mapping. Firstly, to base their work and model on the model that the user creates on their own, or secondly, to give the user the beginnings of a model which they might build upon. The second option is preferable because the designer develops the most accurate and complete model of the system, and also can not be sure that the user will create a suitable model on their own.*

The basis or "germ" for a model is metaphor, or "model of a model" (what Streitz calls a "higher order model", Streitz 1988). The designer should aim towards constructing conceptual models that maximise the possibility of the user forming a useful model by employing a clear, complete, and consistent metaphor in the model.

The suggestion by the Star interface developers to make use of familiar user's conceptual models is echoed by Cox & Walker (1990) who suggest building a conceptual model directly from the task domain.** Such an approach does not seem applicable to abstract and relatively novel domains (such as grammatical design systems). It is currently common for designers to make use of formal (and existing) models around which to design their systems (for example, making use of a desktop metaphor or applications such as "HyperCard"). The problem with this approach is that the complexity of the interface required might be underestimated by the designer, or the existing model might be assumed to be sufficient.

Often, conceptual models are not developed (or refined enough) until a working model is made and experimented with. Even then, the designer might not have created the right (or best) model possible, and might not have explored all of the implications that different *modes of interaction* will have on the functionality, conceptual model, and hence interface. There is a need to look at and investigate the roles and relationships that exist between users and the machines they interface with. The way that users perceive a system when using it (or designers, when building it) needs to be explored. The question that could be asked is, "in what other ways can we relate and interact with computers?".

A fresh start: a new perspective

The word, "perspective", has common use, and is often used to mean: "in true proportion", "a fresh outlook", "a point of view", "a way of seeing something"; it finds its way into phrases such as "taking a new perspective" or "putting something in perspective". Although also used in an artistic setting (to describe spatial relations and dimensions depicted on a flat surface) the word, "perspective" will be taken to mean "a relation of parts, to each other and the whole, in a mental view", or "the appearance of objects to relative position".

A person having a perspective on an issue forms certain opinions about the issue under view; this concept has been developed by Kammersgaard (Kammersgaard 1990) into a means of looking at Human-Computer Interaction (HCI, based on the roles of perspectives in computer science, from the work of Nygaard and Sørgaard 1985). Four archetypal perspectives on computer use have been identified by Kammersgaard - the systems perspective, the dialogue partner perspective, the tool perspective, and the media perspective - and presented as a small taxonomy based upon two fundamental distinctions: whether the focus is on the individual or group use, and whether the focus is on the contents of the interaction or the way the interaction is expressed.

The system perspective views the human as a component of a system (that is, the human is seen as a machine). Collaborative work is supported by the media perspective which views the computer as a "window" between users (as though they can communicate directly to each other). The tool perspective focuses on the computer being a "dumb" (though specialised) tool and the user being able to operate it in the task domain (where they might be considered experts). The dialogue partner perspective regards the computer as having human communicative abilities and is in dialogue with the user on a task.

The tool and the dialogue partner perspectives (both, individual contexts) can be seen in current generative systems. Firstly, the systems support

---

* Users tend to create models that are overly complex (hence difficult to generalise), based on "magical" or superstitious presumptions, or that are overly simple (therefore not providing a good foundation for basing reasoning and model improvement).

** The example given by Cox and Walker (1990) is of a vehicle pooling system, where "car" icons (which might contain registration details) are "dragged" onto a calendar to set the date that the vehicle is required.

direct manipulation and the expression of design rules (that is, the tool perspective), and secondly, they can be viewed as being cooperative partners in the search for designs and the offering of alternate paths to the user. Through the use of each of these perspectives the designer can be focused on the important issues of the interaction, and investigate avenues of functional requirements. A view of the user is given by each perspective and describes many of the cognitive processes that the user might require and employ. For the user, the role they are to play can be elaborated, encouraging them in the way they are to approach and use the system.

The different perspectives bring into focus various issues of the interactive process (as expressed in FIG 1). The tool perspective suggests that the interaction process involves selecting and using a tool on a material in a domain, evaluating the result and repeating the process until a required state is achieved. Direct manipulation and immediate feedback are needed. This means that the media structure must reflect the way in which the material (modelled in the domain representation) is changed by the tool - as a result, altering the operative cognations of the user. The command language must be simple (for direct manipulation) and deals with the selection and operation of the different tools that the computer is shown to have in this perspective. Conceptual models need to be based on one or more metaphors (and typically deal with tools with which the user is already familiar).

The dialogue partner perspective portrays the user in communication with the computer (the partner) to perform a task. Importance is placed on making the computer appear as human as possible. Kammersgaard suggests that this may imply the use of "natural language" as the command language and a media structure that acts as a medium between the partners, which changes only to support the communication between them. As the computer is "assumed" to understand the human "speaking" (for example, typing) in their own language, a conceptual model is not required, because the human only needs to know that they can interact with the system just as they do with a human. (The system might have explanation facilities and identify parts of the task for the user.)

Some systems for Building and Architecture

Kammersgaard sees the dialogue partner perspective as problematic because technology does not fully support the requirements of the interface as stated in his definition. Other researchers differ in this view; for them, the notion of the computer being a partner in design can be seen in a number of architectural design systems:

• "FLATS", a prototype Cooperative Computer-Aided Design system for the design of small architectural floor plans (Kochhar and Friedell 1990) allows the user to control, explore and limit the production of intermediate designs. It controls the combinatorial explosion when generative design rules are applied in an uncontrolled and undirected manner.

• The idea of the computer being an "intelligent assistant" to assist the architectural designer in specialist or mundane tasks is presented by Guéna and Zreik (1989); it can be either an expert (where it solves delicate, complex problems and the human learns from it) or an assistant (where the computer learns, and later executes, some of the tedious tasks that the human performs). The rendering of intelligent advice is at the core of many simulation and evaluation systems. A recent example is the program "ENERGRAPH" (Tao-Kuang Huang et. al. 1992) which supports the design of energy efficient buildings. Intelligent capabilities check the values input by the user for reasonableness, and also suggest ways in which the modelled energy design could be improved.

At the other end of the spectrum, the computer can be seen in its "traditional" role as a tool, but now with an appreciation from the angle of human-computer interaction. A simple grammar interpreter, "DiscoverForm" (Carlson and Woodbury 1991), which allows the user to directly and very simply experience their changes, can be seen purely as a tool, totally under human control. Most commonly (as in the examples above used to illustrate the dialogue partner perspective) we see varying degrees of a tool/dialogue-partner combination, where the computer is a mix of a specialised tool or a partner. "Tartan Worlds", (Woodbury et. al. 1992), a prototype system for the generation of 2-D configurations of graphical symbols on "tartan grids", allows the user to interact with it as a tool, and also presents design options to the user as a partner, proceeding only with their direction.

Extending perspective by analogy

In everyday use, the term "perspective" usually refers to one view, not many, nor in an active and dynamic way. This differs from the notion presented by Kammersgaard, of perspectives being a way of looking at Human-Computer Interaction, and most useful when the different views are shifted between or combined. Perspectives provide a way of viewing the interactive process and a way of considering the relationship between human and computer; the requirements of the interface can be drawn forth, and important areas for investigation stressed. If the user is made aware of the notion of perspective then their role is elaborated, encouraging the way in which they approach and use the system. In a similar vein, most writers on user interfaces suggest that the design of the interface proceed from the user's point of view, that is, that the designer should put themselves in the place of the user when software is being written - in effect, seeing it from "their perspective". This implies thinking like the user, and in this way the designer might build a suitable mental model for them. For design systems, with issues in HCI still needing investigation, and with functionality to explore and develop further,

the study of our relationship when using computers may have major implications for the functionality, conceptual model and hence, interfaces produced.

Kammersgaard suggests that his use of Perspective is only one view when he states "...one can not avoid having a perspective on them" (p.42). I would surmise that Kammersgaard finds the dialogue partner perspective problematic only because of his *own* perspective on the matter. People *do* form different views in HCI: a psychologist may see psychological issues, a graphic artist may see opportunities for artistic design, and a computer scientist may see the technical issues of programming. Looking at things in a different way is beneficial; the essence of discovery in creative thinking. The writer on creative thinking, von Oech, calls this "Getting Whacked", a changing of mind-set which not only stimulates the asking of questions but forces a rethink on problems and a viewing of situations in a different way. A major tool in this generation of new and creative thoughts is the use of analogy.

The concept of analogy is unavoidably drawn into the discussion of perspective, for the ideas of "tools" and "partners" are used to explain issues in interaction. To further explain the concept of perspective one might use analogies dealing with viewing and points of reference. For example, perspectives might be seen as peep-holes into the model of an application. Different users might look through different holes, so that in creating these holes, we (that is, system designers) might target certain users and hence certain uses for our applications. The perspective will not change the object under observation, only our perception of it.

Drawing an analogy with photography, we might imagine people selecting different subjects to view, and composing unique images with different intentions. One is able to zoom in and out of a scene or pan over it for a different content; focal lengths can be altered to make foreground objects step out of the background, and one can "colour" the view by employing tinted lenses.

This "play with analogy" appears to be a logical step in the discussion of perspective. In effect, one is applying a perspective (a different view) to the concept of Perspective as presented by Kammersgaard in his discussion of roles, relationships and interactive processes. The extension of roles by analogy is taking a step away from the issue and looking at it in a more abstract level. As Kammersgaard proposes a shift along one axis in our interpretation of human-computer interaction, I suggest that "taking a perspective on perspectives" hints at the incorporation of a second axis, analogy. This "added dimension" should, I believe, add a richer, more flexible, and ultimately, more useful view.

Kammersgaard stops short of elaborating ways to make use of the idea of perspectives and to generate fresh ideas (other than to see HCI in a new light). Through the augmentation by analogy, perspective could become a tool for conceptual model construction, which may in turn elicit new ideas for functionality, and eventually the interface in which it is presented; out of an environment teaming with Graphical User Interfaces (which suggest that they are a metaphor to be immediately useful), innovative ideas might evolve. It is probable that out of this will emerge:

- new and better metaphors.
- fresh ideas for functional requirements.
- a guiding, top-down principle comparable to the guiding nature provided by remaining consistent with a conceptual model.
- a new appreciation by designers and users for the way that computers can support people in their roles.

The design of computer systems might be made a more creative and playfully explorative process. Beginning with the perspective of the computer as a partner with the user, one might map the relationship analogously to that of a human guide for an explorer. (This of course assumes that the users of the system are familiar with this example. For generative systems, this analogy seems most appropriate.) The situation of an explorer employing a guide may imply that the explorer is in charge of the journey, but accepts advice from the guide (who has a better understanding of the terrain and its complexity). It is the explorer, however, who has an understanding of how their exploration fits into the context of their work. Mentally playing with an analogy suggests the roles we (and the computer) may play, our relationship together and the allocated tasks that each of us performs. The required functionality might be considered from this analogous day-dreaming - the guide might be one to carry provisions or a toolkit of useful items, may explain features on a map, warn us of danger, and give advice or suggest paths to take.

The old, bottom-up approach to the design of many systems, results in user interfaces that grow with unrelated features added together with no coherent techniques. As power and improvements to functionality are added to the system (part of an ongoing system life-cycle) no design theme emerges. (If it does, the user can not always rely on it being consistent.) Many systems-design problems are not encountered until a working version of the system is developed. Due to constraints of time and money, the designer is not granted many chances to get the interface right. Clearly, a top-down approach is needed.

The Xerox Star designers have shown that conceptual models are unifying elements of an interface that help maintain consistency and support learnability. Perspectives extended by analogy could be used to ensure that other beneficial guidelines are incorporated into the description laid out by the conceptual model. For example, Nievergelt's criteria for good, interactive software (Nievergelt and Weydert 1980), which is given by the questions:

- Where am I?
- What can I do here?
- How did I get here?
- Where can I go and how can I get there?

might make use of the metaphor of a journey. To continue the example of the explorer, the guide would be the one to know the answers to these user needs.

Conclusion

That computer systems will continue to increase in power and complexity should not be in dispute; it is in providing a clear and useful interface to the user that is the most complex issue. Making use of powerful environments (for example, object-oriented systems, prototyping systems or various user-programmable applications) will not ensure that a useful model has been presented to the user. It is what is done with these environments that it important - requiring creativity, which is a topic also central to with the practice of architecture and engineering.

Interface design problems should be looked at from every possible angle prior to development and may prevent designers from overlooking important issues. "Chewing over" an interface problem (like a period of incubation during the creative process) may result in the "creative leap" needed to solve the problem or bring something new to it.

The ideas presented in this paper have not been about producing solutions but about stimulating thinking; it is hoped that the use of perspective extended by analogy might provide a stimulus to rethink and explore, not necessarily providing a map. As "...even the most experienced researchers in this area [HCI] are pioneers" (Hicks and Essinger 1991, p.1), perhaps the higher-level approach suggested here might result in an interesting find on that journey of exploration.

References

Carlson, C. and Woodbury R.F. 1991. "Hands-on exploration of recursive forms." Technical Report, Department of Architecture, Carnegie Mellon University, Pittsburgh, Pa, 1991.

Cox, K., and Walker, D. 1990. *User-interface Design.* Advanced Education Software, Hackett, ACT, Aust.

Guéna, F. and Zreik, K. 1989. "An architect assisted architectural design system." In *Artificial Intelligence in Design,* J. S. Gero, ed. Springer-Verlag, Berlin, 159-179.

Hicks, R. and Essinger, J. 1991. *Making computers more human: designing for human computer interaction.* Elsevier Science Publishers, Oxford.

Huang, T.; Degelman, L.O.; and Larsen, T.R. 1992. "A visualization model for computerized energy evaluation during the conceptual design stage (ENERGRAPH)." In *Computer Supported Design in Architecture: mission, method, madness,* K.M. Kensek and D. Noble, eds. The Association for Computer Aided Design in Architecture, Uni of Southern California, 195-206.

Kammersgaard, J. 1990. "Four different perspectives on human-computer interaction." In *Human-Computer Interaction,* J. Preece and L. Keller, eds. Prentice Hall, London, 42-63.

Kochhar, S. and Friedell, M. 1990. "User control in cooperative computer-aided design." In *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology* (Snowbird, Utah, USA, October 3-5 1990) ACM Press, NY., 143-151.

Licklider, J.C.R. 1960. "Man-computer symbiosis" *IRE Transactions on Human Factors in Electronics* HFE -8(1), 4-11.

Nievergelt, J. and Weydert, J. 1980. "Sites, modes and trails: telling the user of an interactive system where he is, what he can do, and how to get places." In *Methodology of Interaction,* R.A. Guedj, P.J.W. ten Hagen, F.R.A. Hopgood, H.A. Tucker, and D.A. Duce, eds. North Holland, Amsterdam.

Nygaard, K. and Sørgaard, P. 1985. "The perspective concept in informatics." In *Computers and Democracy,* G. Bjerknis, P. Ehn and M. Kyng, eds. Avebury, Aldershot, 371-394.

Singleton, W. T. 1974. *Man-machine systems.* Penguin, Harmondsworth.

Smith, D. C.; Irby, C.; Kimball, R.; Verplank, B.; and Harslem, E. 1982. "Designing the Star user interface." *Byte* 7(4), 242-282.

Streitz, N.A. 1988. "Mental models and metaphors: implications for the design of adaptive user-system interfaces." In *Learning issues for intelligent tutoring systems,* H. Mandl and A. Lesgold, eds. Springer-Verlag, New York.

von Oech, R. 1983. *A whack on the side of the head.* Warner Books, New York.

Winograd, T. and Flores, C.F. 1985. *Understanding computers and cognition: a new foundation for design.* Abex, Norwood, New Jersey.

Woodbury, R.W.; Radford, A.D.; Taplin, P.N.; and Coppins, S.A. 1992. "Tartan Worlds: a generative symbol grammar system." In *Computer Supported Design in Architecture: mission, method, madness,* K.M. Kensek and D. Noble, eds. The Association for Computer Aided Design in Architecture, Uni of Southern California, 221-220.

Wozny, L.A. 1990. "The application of metaphor, analogy, and conceptual models in computer systems." *Interacting with Computers* 2(3), 273-283.