

Computer-Aided Design Systems for the 21st Century: Some Design Guidelines

Jens Pohl, Art Chapman, and Kym Jason Pohl

Collaborative Agent Design (CAD) Research Center
Cal Poly State University
San Luis Obispo, California, USA

ABSTRACT

This paper proposes nine design principles for a new generation of computer-aided design (CAD) systems that actively support the decision making and problem solving activities of environmental design. Foremost among these are: a meaningful internal object-based representation of the artifact being designed within its environmental context; a collaborative problem solving paradigm in which the human designer and the computer form a complementary partnership; and, the notion of decision-support tools rather than predefined solutions. Two prototype computer-aided design systems implemented by the CAD Research Center that embody most of these concepts are described. ICADS (Intelligent Computer-Aided Design System) incorporates multiple expert agents in domains such as natural and artificial lighting, noise control, structural system selection, climatic determinants, and energy conservation. Given a particular building design context, the agents in ICADS draw upon their own expertise and several knowledgebases as they monitor the actions of the human designer and collaborate opportunistically. KOALA (Knowledge-Based Object-Agent Collaboration) builds on the multi-agent concepts embodied in ICADS by the addition of two kinds of agents. Mentor agents represent the interests of selected objects within the ontology of the design environment. In the implemented KOALA system building spaces are represented by agents capable of collaborating with each other, with domain agents for the provision of expert services, and with the human designer. Facilitator agents listen in on the communications among mentor agents to detect conflicts and moderate arguments. While both of these prototype systems are limited in scope by focussing on the earliest design stages and restricted in their understanding of the inherent complexity of a design state, they nevertheless promise a paradigm shift in computer-aided design.

THE DESIGN ACTIVITY

Design is indeed an ubiquitous activity. In the physical world every artifact, whether it be a coffee maker, a miniature silicon sensor for invasive blood pressure monitoring, an automobile, or a building, is the result of some kind of design activity. However, design is concerned not only with the creation of artifacts. Any problem solving situation in which there exists an element of the unknown, such as lack of information or incomplete knowledge of the relationships among issues, involves an intellectual effort that can be categorized as design (Simon 1996).

It follows that design is more than the rote calculation of an algorithm or the copying of a known process, although either or both of these may be useful design tools. For an intellectual endeavor to qualify as a design activity it must involve relationships that cannot be totally defined and

appear to pose some degree of conflict. Indeed, the resolution of actual, perceived and potential conflicts is a fundamental ingredient of all design endeavors (Pohl et al. 1994).

Typically, design requires decisions to be made among several imperfect alternatives. It is in the nature of those decisions that designers will often find the need to supplement logical reasoning with intuitive feelings about the problem situation that can lead to creative solutions and new knowledge. As a rule such new knowledge cannot be logically deduced from the existing available knowledge and is validated only after the solution has been discovered and tested. In this respect design is not unlike the decision making activities that occur in a wide range of complex problem situations that have to be dealt with in many professional fields such as management, economics, medicine, law, transportation planning, and military command and control.

GUIDING PRINCIPLES FOR THE DESIGN OF COMPUTER-AIDED DESIGN SYSTEMS

Over the past decade the work of the Collaborative Agent Design (CAD) Research Center has focused on complex problems and computer-based decision-support systems that are designed to assist, not replace, human decision makers. Our experience has confirmed that the relative level of complexity of a problem is primarily a function of the number and strengths of the inter and intra relationships that exist among internal and external components of the problem, and the degree of uncertainty that surrounds the definition of these components.

The systems that we have developed are in several respects quite different from currently available commercial computer-aided drawing and modeling systems. These differences are based on the following principles that we have established over the years.

Principle 1: Emphasis on Partnership

A successful computer-aided design or decision-support system is one that assists rather than replaces the human decision maker. Human beings and computers are complementary in many respects. The strengths of human decision makers in the areas of conceptualization, intuition and creativity are the weaknesses of the computer. Conversely, the strengths of the computer in computation speed, parallelism, accuracy and the persistent storage of almost unlimited detailed information are human weaknesses. It therefore makes a great deal of sense to view a decision-support system as a partnership between human and computer-based resources and capabilities. Automation should be restricted to the monitoring of problem solving activities, the detection of conflicts, and the execution of evaluation, search and planning sequences.

In this partnership a high level of interaction between the user and the computer is a necessary feature of the decision-support environment. It provides opportunities for the user to guide the computer in those areas of the decision making process, such as conceptualization and intuition,

where the skills of the user are likely to be far superior to those of the computer. Particularly prominent among these areas are conflict resolution and risk assessment.

Principle 2: Collaborative and Distributed

Design, or complex problem environments in general, normally involve many parties that collaborate from widely distributed geographical locations and utilize information resources that are equally dispersed. The decision-support system can take advantage of the distributed participation by itself assuming a distributed architecture. Such an architecture typically consists of several components that can execute on more than one computer. Both the information flow among these components and the computing power required to support the system as a whole can be decentralized. This greatly reduces the potential for communication bottlenecks and increases the computation speed through parallelism.

Another advantage of the distributed approach is the ability to modify some components of the system while the system as a whole continues to operate with the remaining components. Similarly, the malfunction or complete failure of one component does not necessarily jeopardize the entire system. This is not so much a matter of redundancy, although the distributed architecture lends itself to the provision of a high degree of redundancy, but rather a direct result of the physical independence of the components. While the components may be closely integrated from a logical point of view they can operate in their own autonomous physical environment.

Principle 3: An Open Architecture

The high degree of uncertainty that pervades complex problem environments extends beyond the decision making activity of the collaborating problem solvers to the configuration of the decision-support system itself. The components of the system are likely to change over time, through modification, replacement, deletion and extension. It should be possible to implement these changes in a seamless fashion through common application programming interfaces and shared databases.

Principle 4: Tools, not Solutions

The decision-support system should be designed as a set of tools rather than as solutions to a predetermined set of problems. The indeterminate nature of design problems does not allow us to predict, with any degree of certainty, either the specific circumstances of a future problem situation or the precise terms of the solution. Under these circumstances it is far more constructive to provide tools that will extend the capabilities of the human decision maker in a highly interactive problem solving environment.

In this sense a tool is defined more broadly than a sequence of algorithms, heuristics or procedures that are applied largely on the direction of a user. Tools can be self-activating, be capable of at least semi-autonomous behavior, and cooperate with each other and users in requesting and providing services.

Principle 5: High Level Internal Representation

The ability of a decision-support system to have some level of understanding of the meaning of the information it processes is the single most important prerequisite for a cooperative and collaborative problem solving environment. A high level representation of the real world objects that define the problem system forms the basis of the interactions between the users and the system and, also, the degree of intelligence that can be embedded in its components. For example, the geometric descriptions created by current CAD systems are essentially limited to points, lines, polygons, faces, and primitive geometric solids. What is needed is a comprehensive description of the artifact in terms of objects that have meaning in the real world in which the artifact will be fabricated, assembled and used.

While it is quite difficult to generate higher level information from lower level data, the reverse is generally fairly straightforward. Given the high level knowledge that an object is a window in the external wall of a room, it is relatively easy to generate the daylight illumination characteristics at any point in the room based on the higher level description and a few default assumptions. To determine on the basis of polygons and surface parameters that the object is a window with particular characteristics (e.g., orientation and glazing) is a much more difficult undertaking (Pohl 1995). To the designer a building consists of real world objects, such as rooms, walls, windows, doors, structural components, furniture, and so on. Each of these objects has attributes and relationships that determine its behavior under certain conditions. These semantic descriptors form the basis of collaboration among human problem solvers, and are likewise the fundamental unit of communication in a computer-aided design environment.

Principle 6: Embedded Knowledge

The computer-aided design system should be a knowledge-based system. In this context knowledge can be described as experience derived from observation and interpretation of past events or phenomena, and the application of methods to past situations. Knowledge-bases capture this experience in the form of rules, case studies, standard practices, and typical descriptions of objects and object systems that can serve as prototypes. Problem solvers typically manipulate these prototypes through adaptation, refinement, mutation, analogy, and combination, as they apply them to the solution of current problems (Gero et al. 1988).

Principle 7: Decentralized Decision Making

The decision-support system need not, and should not, exercise centralized control over the decision making environment. Much of the decision making activity can be localized. For example, components of the system (e.g., mentor-agents) that are responsible for pursuing the interests of real world objects, such as spaces and similar building elements in architectural design (Pohl 1996) and management personnel in commercial and industrial applications (Pan and Tenenbaum 1991), can achieve many of their objectives through service requests and negotiations that involve only a few nodes of the problem system. This greatly reduces the

propensity for the formation of communication bottlenecks and at the same time increases the amount of parallel activity in the system.

The ability to combine in a computer-aided design system many types of semi-autonomous and autonomous components (i.e., agents), representing a wide range of interests and incorporating different kinds of knowledge and capabilities, provides the system with a great deal of versatility and potential for problem solving to occur simultaneously at several levels of granularity. This is similar to human problem solving teams in which individual team members work concurrently on different aspects of the problem and communicate in pairs and small groups as they gather information and explore sub-problems.

Principle 8: Emphasis on Conflict Identification

The decision-support system should focus on the identification rather than the automatic resolution of conflicts. This notion gains in importance as the level of complexity of the problem system increases. The resolution of even mundane conflicts can provide subtle opportunities for advancing toward design solution objectives. These opportunities are more likely to be recognized by a human designer than a computer-based agent. The identification of conflicts is by no means a trivial undertaking. It includes not only the ability to recognize that a conflict actually exists, but also the determination of the kind of conflict and the relationships that appear to have precipitated the conflict. Tracing these relationships may produce more progress toward a solution than the resolution of the conflict itself.

Principle 9: The Computer-User Interface

The importance of a high degree of interaction between the designer(s) and the various components of a computer-aided design system is integral to most of the principles described here. This interaction is facilitated by two system characteristics: a high level object representation; and, an intuitive user interface. The user interface should be graphical in nature. The human cognitive system excels in pattern matching. Words and numbers require the performance of a translation task that is relatively time consuming, subject to information loss, and carries with it the potential for confusion and misinterpretation.

An on-line help system should be available to both assist the user in the execution of operational sequences and provide explanations of system activities. The latter should include exploration of the recommendations, evaluation results and proposals contributed by the various components (e.g., agents) of the system.

THE ICADS PROTOTYPE

The Intelligent Computer-Assisted Design System (ICADS) provides a collaborative building design environment in which multiple expert agents, referred to as Intelligent Design Tools

(IDTs), opportunistically communicate with each other and the human designer in their quest to assist in the decision making process (Pohl 1991). Emphasis has been placed on the earliest design stages during which conceptual solution strategies are formulated and the framework for the entire design process is established. Poor judgement, misinterpretation of data and unavailability of relevant information during this conceptual stage often lead to fundamental design errors that are difficult and costly to correct downstream.

The CAD Research Center was fortunate to obtain permission from Accugraph Corporation (El Paso, Texas) to incorporate its MountainTop computer-aided drawing package in the ICADS working model. Minor modifications were made to this commercially available CAD system to provide direct access to the data structure of the drawing currently displayed on the screen.

A schematic diagram of the implementation architecture of the ICADS model is shown in Fig.1. It includes two knowledge bases, several sources of reference data, a CAD drawing system, a multi-function user interface (i.e., Design Interface), and an Expert Design Advisor that contains eight domain experts (i.e., IDTs), and a blackboard coordination expert (Pohl and Myers 1997).

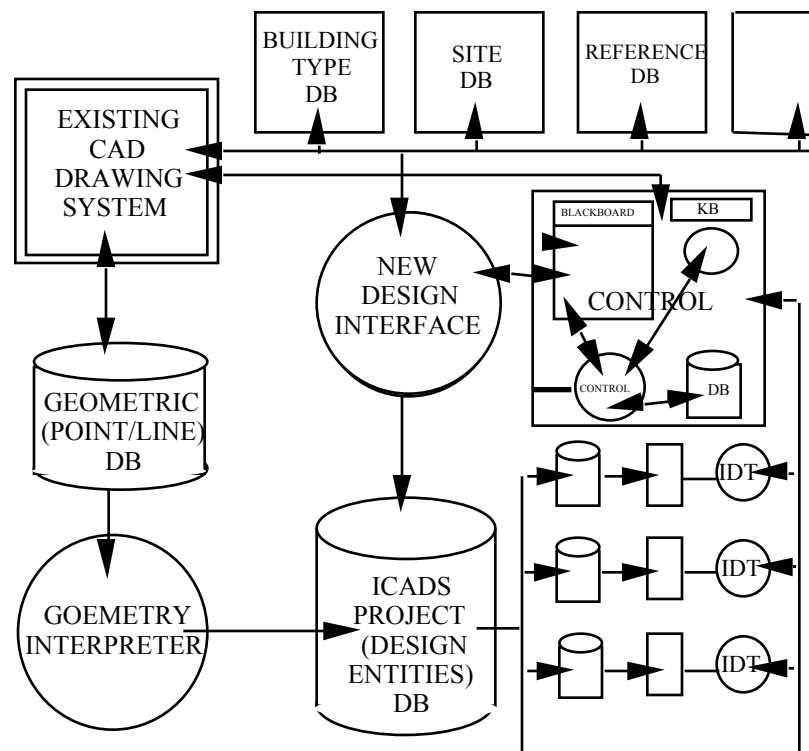


Fig.1: Schematic architecture of the ICADS model

The scope of the implementation environment has been restricted in terms of both the breadth of information available to the designer and the range of design functions supported. A subset of the domain experts necessary for a complete design environment has been developed and a specific design project scenario has been selected for demonstration and testing purposes. The domain experts, shown as IDTs in Fig.1 and as small agent status windows on the left side of the ICADS screen shown in Fig.2, represent sub-areas within the structural, thermal, sound, daylighting, artificial lighting, energy conservation, building massing, and cost estimating domains of the architectural design application.

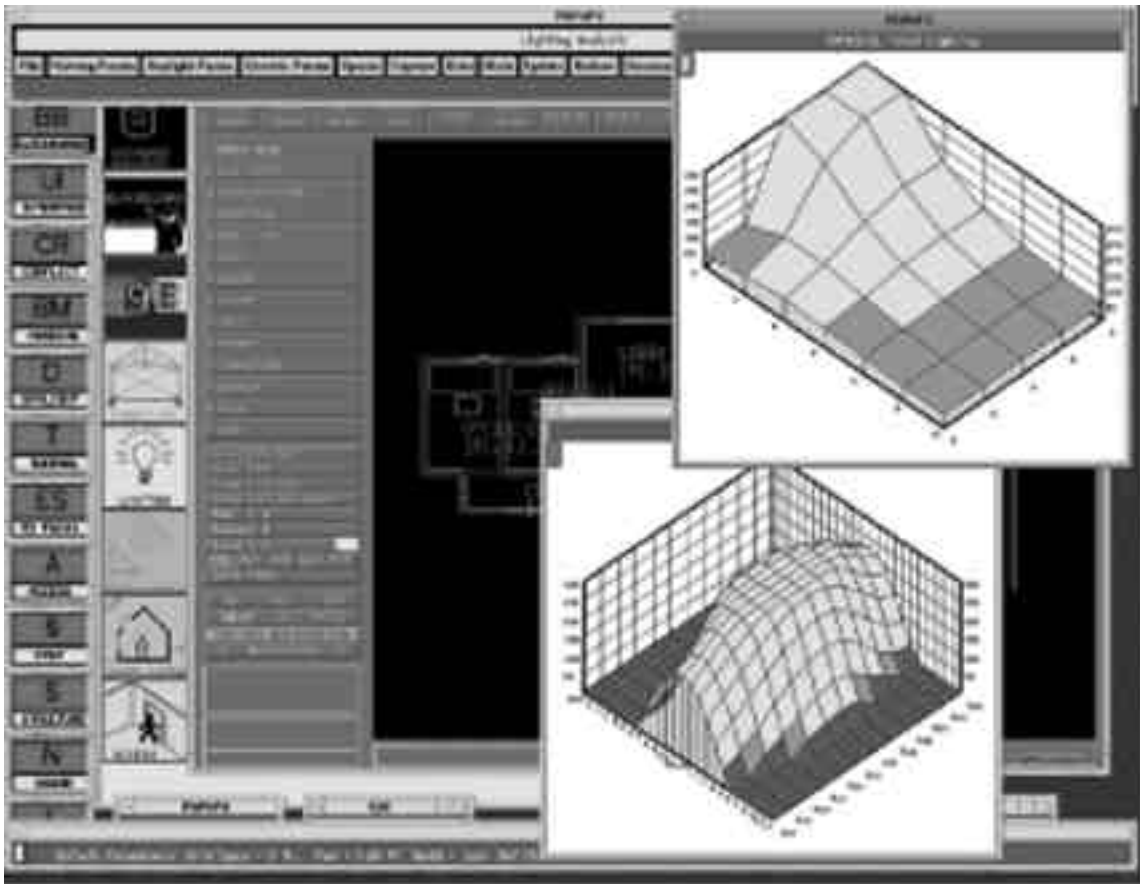


Fig.2: ICADS screen with a daylighting exploration in progress

The test project is the design of a community center building for a small city on the central coast of California, Pismo Beach. The information resources provided by the working model are more general in nature and include Building Type and Site/Neighborhood knowledge bases, as well as several Reference databases containing material and constructional information.

The Expert Design Advisor is responsible for the evaluation of the evolving design solution and the resolution of conflicts that may arise when solution proposals in one domain interfere with solutions in another domain. It comprises advisory components, a coordination expert and operational components, as shown below:

Advisory components: Structural domain expert
Lighting domain experts
Sound domain expert
Thermal domain expert
Energy conservation expert
Building massing (footprint) expert
Construction costs expert

Coordination expert: Message Router
Conflict Resolver
Geometry Interpreter

Operational components: Attribute Loader
Semantic Network of frames

The coordination mechanism used by the ICADS model is a data-blackboard. The ICADS Blackboard differs from earlier blackboard implementations by others, in that it is not used for planning and scheduling activities (Durrant-Whyte 1988, Hayes-Roth et al.1986). Instead the agents are allowed to execute immediately and in parallel on distinct processors, as values appropriate to them are posted on the Blackboard.

The Blackboard Control System

The essential notion of the ICADS Blackboard is the centralization of all of the facts that are shared by the components that it controls. There are two kinds of facts on the Blackboard, namely: the ***current values*** of solution parameters that describe the state of the design solution; and, communication or ***current advice facts***. The latter provide communication among the components that use the Blackboard, but are not accepted as part of the solution set for the design. In particular, the output of IDTs is advisory and is placed on the Blackboard so that the advice can be taken by other components in the system. The decision to have the advisory components that respond to changes in ***current value facts*** write their results on the Blackboard assures overall system control. The IDTs react to these facts and produce ***current advice facts***. Both of these kinds of facts are used by the Blackboard Control Expert to update the ***current value facts*** that describe the current state of the design solution.

However, ***current value facts*** describing the geometry of the solution model are also produced by the Geometry Interpreter. Since only the Control Expert (i.e., the Conflict Resolver) is permitted to post ***current value facts*** on the Blackboard and the advisory components react to

changes in these facts, the operation of the system as a whole is governed by the rules of the Control Expert. With the exception of the Geometry Interpreter, these rules are written in the CLIPS expert system shell language (NASA 1989) to ensure a high degree of clarity in describing the control function and provide a rapid prototyping capability for development purposes. It must be noted that the Geometry Interpreter is considered to be part of the Blackboard Control Expert because drawing changes must be reflected in the Blackboard through *current value facts*.

The set of CLIPS rules that comprise the Control Expert are responsible for writing all other *current value facts* on the Blackboard. These rules drive the remaining operational functions of the system. It must be emphasized that both the Geometry Interpreter and Control Expert rules respond also to mouse and keyboard actions to preserve the ability of the designer to orchestrate the design process. IDTs may only recommend changes to the current state of the design solution by writing their results into private areas of the Blackboard, where they are examined by the Control Expert. Accordingly, there are private and global areas in the Blackboard. The private areas hold the *current advisory facts* to provide input to the Blackboard Control Expert. The global areas provide the accepted, or arbitrated, *current value facts* that can be freely accessed by any advisory component.

The Semantic Network Implementation

The solution model that evolves out of the architectural design process can be viewed as an assembly of components or objects that have both physical form and functional relationships. In architecture such objects include buildings, floors, spaces, walls, openings and furniture. Typically, each object represents an hierarchy of lower level objects, and is at the same time part of the hierarchy of one or more higher level objects. In ICADS these objects are implemented in a frame-based representation as a network of *project design* and *solution design* object frames. This network serves as the nerve center of the Blackboard in supplying information related to the current state of the design solution to the IDTs and the Conflict Resolver.

The *project design* object frames represent the current state of the design problem context. They are established by the Attribute Loader from information contained in the Building Type and Site/Neighborhood knowledge bases. In the current version of ICADS these *project design* object frames are initialized at the beginning of the design session, since the designer is restricted to read-only access of these knowledge bases. A future more comprehensive ICADS implementation should include the ability of the Attribute Loader to refresh individual frames whenever the designer modifies information in the knowledge bases.

The *solution design* object frames are dynamically established by the Geometry Interpreter and represent the geometric description and relationships, and the evaluation status of the current solution model. Separate frames are created for each design object included in the currently loaded drawing, and the results generated by the IDTs and the Conflict Resolver are posted to these frames and their associated frames (i.e., *relation* frames) as additional slot values.

The Attribute Loader

The Attribute Loader is the interface between the design knowledge bases (i.e., Building Type and Site-Neighborhood) and the *project design* object frames of the semantic network. For each design object, the Attribute Loader accesses these knowledge bases (i.e., relational database) and loads values for the non-geometric attributes of that object. While most of these values are transferred directly from the database to frame slots, some are inferred from several database values. The information loaded into the *project design* object frames constitutes the prototype knowledge and site specific context information available to the designer during the development of the solution model.

The Attribute Loader is written in CLIPS as a set of production rules and undertakes its loading task in the following sequence:

- Requests the Blackboard at the beginning of the initialization process to inform it whenever a *project design* object frame is added to the *solution design* object portion of the network.
- On receiving notice of a new *project design* object frame the Attribute Loader accesses the appropriate SQL database tables using a CLIPS-SQL query function, and asserts the returned values directly into the CLIPS fact-list of the Attribute Loader.
- Infers those values that are derived from two or more database values.
- Creates all default values that are present in the Attribute Loader as preset facts.
- Transfers all required values from its own fact-list to the appropriate slots of the *project design* object frames.

It should be noted that the *project design* object frames are actually created by the Attribute Loader. In this respect, the Attribute Loader and the Geometry Interpreter are the only components of the Blackboard Control System that are able to create frames in the network. The Geometry Interpreter establishes the *solution design* object frames as it analyzes the drawing currently loaded in the CAD Drawing System. Since its mode of operation is based on the identification of design objects (e.g., spaces, walls, etc.) these same objects become frames in the network. The Attribute Loader establishes the *project design* object frames from the prototype information contained in the design knowledge bases. However, since it is activated by the Blackboard in response to the creation of frames by the Geometry Interpreter, one frame for each design object type (i.e., space, wall, etc.) will be included in the *project design* object section of the semantic network.

The Conflict Resolver

The Blackboard Control Expert is implemented as the Conflict Resolver. Its principal purpose is to assert *current value facts* frame slots, representing the current state of the evaluation process performed by the IDTs, onto the semantic network resident in the Blackboard. To accomplish

this, the Conflict Resolver requests from the Message Router all of the *solution design* object frames that contain results generated by the IDTs. The Conflict Resolver is the only expert allowed to request the results of the individual IDTs. Part of the fact-list of the Conflict Resolver then represents those private areas of the Blackboard in which the IDTs propose possible solution values.

ICADS supports three categories of *current value facts*: values that result from solutions proposed by a single IDT; values that result from solutions proposed by several IDTs for a common current value; and, values that must be inferred from solutions proposed by several IDTs. In the first category, which represents solution values unique to a single IDT, the Conflict Resolver does not change the values proposed by the IDT. The proposed solution values are simply posted as current values into the appropriate frame slots. In the second category two or more IDTs propose differing values for the same solution parameter. In such direct conflict situations it is the responsibility of the Conflict Resolver to either determine which of the values is better or to derive a compromise value. Of course, the process of resolution may require the Conflict Resolver to change several *current value facts* in addition to those in direct conflict.

In the third category the Conflict Resolver deals with proposed solution values that are indirectly in conflict with other proposed solution values and current values. The resolution rules for this category allow the Conflict Resolver to make the necessary modifications to any of the values involved. Under these conditions, in addition to changing proposed solution values the Conflict Resolver may also change *current value facts*.

The Message Router

The Message Router is the central communication hub in the ICADS Blackboard. It communicates with the other processes via sockets and message queues, and allows for the passing of information in and out of the *solution design* object frames of the semantic network. All information is passed through the Message Router.

During the first step of the initialization process that follows the loading of a drawing into the CAD Drawing System, the Message Router receives from each IDT a list of the attributes for which it requires *project design* and *solution design* object frame slot values. These frame slot needs of the IDTs become the core of the Message Router's fact-list. Having received the *project design* object frames from the Attribute Loader the Message Router sends to each IDT the requested values. On completion of the initialization sequence the Message Router removes the *project design* object frames from its fact-list.

When the design process begins the Message Router receives *solution design* object frames from the Geometry Interpreter. As the frames are received the appropriate slot values are passed to the IDTs and the frames are removed from the Message Router's fact-list. Once the IDTs have sufficient information to commence their evaluation process, they begin to send their results in the form of *solution design* object frame slot values to the Message Router. These values are sent onto the Conflict Resolver, allowing the frames to be again deleted from the fact-list of the Message Router. In this way the Conflict Resolver sees all of the suggestions that are being made

for changes in the current values. In other words, the Conflict Resolver has access to both the private and global solution frames in the semantic network.

Implementation and Discussion Issues

Three primary areas of concern have been addressed in the implementation of the Conflict Resolver: control of the number of iterations; the timeliness of the conflict resolution process; and, the quality of any specific conflict resolution. The first concern of the Conflict Resolver is to prevent the Expert Design Advisor from entering into an endless argument state that may arise when two IDTs always return with conflicting values for the same solution parameter. For instance, the Structural IDT may insist for good reasons that the roof construction system should be timber. For reasons that are different but just as persuasive, the Thermal IDT may be unwilling to deviate from its original proposal of a concrete roof system. The Conflict Resolver monitors this type of situation by checking for cycles in the current values and either intervenes based on its own set of conflict resolution rules or notifies the user of the apparently unresolvable conflict.

Another concern is the potential for a cascading condition that may arise when a minor change by an IDT causes a major recomputation. In the current version of ICADS the possibility of such a condition occurring is exacerbated because IDTs will fire on any change to a current value and the Conflict Resolver will respond to any proposed changes posted by the IDTs. An attempt has been made to foresee events that could conceivably lead to this undesirable condition. Where appropriate, tests have been included in IDT rules to determine whether the current divergence between the most recent IDT result and the corresponding current value (proposed by the Conflict Resolver) is sufficiently large to warrant further action by the IDT. In respect to the issue of timeliness the Conflict Resolver will not post a current value to the Blackboard until it has seen all of the *solution design* object frames that are involved in a given conflict.

Although this work commenced a decade ago in the late 1980s (Pohl et al. 1989) and the focus of the Collaborative Agent Design (CAD) Research Center has since that time shifted decidedly from engineering design to decision-support systems for military and disaster relief applications, ICADS still represents a significant advance over current commercially available computer-aided design systems. It is indeed unfortunate that the fragmented nature of the architecture, engineering and construction (AEC) industry will not allow it to support a concerted development effort in the rapidly emerging area of intelligent decision-support systems. The attendant lack of visionary resolve and scarcity of funding within this industry and the related design professions has forced the mainstream of research and development efforts in this field to become focused mostly on military applications. In the light of this situation the CAD Research Center is particularly grateful to IBM Corporation, Accugraph Corporation, and the US Department of Energy, who supported our initial research efforts in computer-assisted design during the period 1988 to 1994.

THE KOALA PROTOTYPE

The Knowledge-Based Object-Agent Collaborative (KOALA) system attempts to create a design-assistance environment in which elements of the evolving artifact actively participate in the development of the design solution. Similar to the ICADS model, assistance is provided by collections of expert agents and is based on prototypical knowledge. However, the KOALA design environment enhances such functionality by extending the notion of an agent and presenting a somewhat more sophisticated collaborative model involving four agent types; namely: the Designer agent, Domain agents; Space agents; and, Monitor agents.

The Designer Agent

The agent taxonomy begins with identifying the human designer (i.e., the user) as the most capable agent in the computer-based design environment. Unique to this agent is the notion of intent. Intent refers to the goals and objectives of the designer. It is the responsibility of this agent to not only acquire the designer's intent, but to also maintain its reflection in the decisions being made by the agents in the system. Intent may be explicitly expressed in the form of design criteria, such as performance requirements, or implicitly hidden in decisions that are influenced by vaguely defined perceptions and subtle nuances. In the design activity, the notion of intent is essentially embedded in the strategy employed by the designer. Unfortunately, this implicit notion is not readily identifiable by observers or, at times, by the human designer as the initiator (Schön 1988).

The current implementation of the Designer agent in KOALA is rather primitive. It is confined to the comparison of design actions by the user and alternative design solution suggestions developed by the agents, with explicitly stated design criteria and specifications. The latter are represented as rules and facts within the knowledge base and fact list of the agent, respectively. Future extensions of the Designer agent will need to explore low level learning capabilities pursuing strategies recently described by Pan (2000).

The Domain Agents

The second category within KOALA's agent taxonomy is the Domain Agent. Service-oriented, Domain agents provide expertise within specific domains of knowledge. Each agent provides expert evaluation and consultation based on its particular domain. Such analysis is largely driven by prototypical knowledge. In other words, the various attributes and characteristics comprising the current solution are compared with those commonly associated with design elements of a similar nature in a related environment. The exact set and depth of domains represented depends on the context in which the application is to be employed.

The initial set of Domain agents represents deeper knowledge in the areas of daylighting, thermal design determinants, noise control, and functional purpose. This limited set could be readily supplemented to eventually extend to at least the range of expertise currently represented in the ICADS agent community.

The Space Agents

The next division within the taxonomy is the Space Agent. Space agents are a specific instantiation of a more general type of agent that can represent the interests of a high level object which plays a significant role in the decision making process of the application environment. Spaces or rooms play an important role in the development of floor plans. The architect manipulates spaces as complex data objects with strong relationships to each other and equally important relationships to data entities that are related but substantially different in nature (e.g., occupant activities, privacy, security, etc.). The ability of the human designer to reason about the relationships among complex data objects is an essential part of the decision making process that underlies the design activity. Computer-assisted design systems, such as ICADS, utilize some form of semantic modeling approach (Myers et al. 1993) to define a common vocabulary that serves as an internal high level representation of real world objects, such as spaces, walls, and opening. This approach provides a workable basis for Domain agents to monitor the evolving design solution and communicate with each other and the designer through some type of coordination facility. The success of this approach must rely heavily on predefined knowledge that is embedded in the agents, and user interactions (i.e., the intervention of the user to maintain and prioritize relationships as a reflection of his/her design intent).

A different approach is to treat the objects that play a major role in the problem environment (e.g., building design), not as passive data entities, but as active agents. Such object-agents can utilize communication capabilities to dynamically create and maintain relationships to other object-agents. Potentially, this would appear to be a significantly more promising approach, since this allows a complex problem system to be decomposed into sub-problems without diluting or losing relationships. To the contrary, relationships are greatly strengthened through the dynamic nature of communication in a collaborative environment. Space agents then are object-agents that have knowledge of their own nature (i.e., essentially the same descriptions that are contained in a 'space' data-object) and the ability to interact with other agents through their communication capabilities. They can act on their knowledge, gain additional information, and request services from other agents.

Prototypical knowledge relative to a space is utilized by a Space Agent to form a set of interests and desires. With each addition of a space into the evolving design, a Space Agent is created and associated with that space. The sole purpose of a Space Agent is to represent the interests of its space. Consequently, each Space Agent views the world (e.g., solution space) from its own, potentially biased perspective. Such biases are an important ingredient of an autonomous environment. As in human group collaborations they reflect the variety of viewpoints that can apply in a given context, and must therefore not be suppressed in the computer-assisted environment. Extensively analyzed, argued, and negotiated, these viewpoints lead to a more comprehensive understanding of the problem and presumably a higher quality solution.

The Monitor Agents

In any highly collaborative environment there is a need for facilitators to detect conflicts and moderate arguments among object-agents. In the ICADS model non-convergence (i.e., the inability of the agents to come to a consensus) could be controlled through various techniques, such as user interaction and the assignment of priorities. In the KOALA system this problem is much more serious, not only because of the relatively large number of object-agents (i.e., Space agents) but also because of the different viewpoints that these agents represent.

For this reason the concept of Monitor agents has been introduced in the KOALA system. The purpose of a Monitor agent is to identify possible conflicts and assist in their resolution through the application of moderating techniques that have been successful in human collaborations. Before identifying these techniques and discussing their application by Monitor agents, it is necessary to consider agent collaboration behavior in more detail.

When a modification to the progressing design occurs, each affected Space Agent will formulate a supporting set of design decisions based on individual constraints and interests. These decisions may include a new building material or structural system, and are presented to the other agents with the intent of achieving global acceptance. As a result, each agent gains exposure to various alternative solutions. However, due to their autonomous nature, Space agents tend to lobby only for outcomes that best satisfy their particular interests. In other words, if left to their own devices, Space agents are reluctant to accept anything that offers a less than perfect outcome from their perspective. Inevitably, this stubbornness may lead to deliberations reaching a stalemate. In such an event, a Monitor agent enters the collaboration as a third party facilitator. The goal of this agent is to bring about agreement through assisting the agents in maintaining clarity and focus with respect to the relevant issue(s). To assist in this task, Monitor agents have several strategies to apply among the deliberating agents; namely the *persuasive strategy*, the *imposition strategy*, and the *user-directed* strategy.

The *persuasive strategy* attempts to use persuasion as a means of achieving global consensus. In essence, the Monitor agent attempts to persuade one or more agents to reevaluate previously unacceptable solutions based on a more flexible heuristic. If this reevaluation leads to global consensus, the common design decision is adopted by the agents and reflected throughout the system. If the agents find suitability with multiple possible solutions, the Monitor agent selects the solution that requires the least amount of loss for the agents.

The *imposition strategy* represents a more forceful approach by attempting to bring about global consensus through compromise. Utilizing this strategy, the Monitor agent searches for a solution to essentially impose on the deliberating agents. However, this solution is by no means arbitrary. Rather, the solution is not only a product of the initial agent deliberation, but it may in fact already be held favorably by a number of the agents. In determining which solution to select, the mediating Monitor agent searches for a majority opinion.

The *user-directed strategy* is the last resort available to KOALA and involves the user as the definitive mediator. The Monitor agent initiates a dialog with the user presenting the particular dilemma at hand. In addition, the Monitor Agent provides the user with a description of the various solutions as presented by the deliberating agents. Based on this information, it is the task

of the designer to decide on the most appropriate solution. However, the user is by no means confined to the solutions proposed by the agents. At any time during the design activity, the designer is free to explore any number of alternative solutions through direct collaboration with the agents. The user may select a subset of Domain and Space agents with which to engage in an exploratory discussion of alternatives to the temporary exclusion of the other agents.

Unique among other design assistance applications, such as ICADS (Pohl et al.1992), the KOALA system combines service-oriented Domain agents, self-motivating Space agents, and facilitating Monitor agents into an abstract agent world. Each of these agents plays a specific role in providing an assistance intensive design environment. The result is a highly collaborative, dynamic model where agents deliberate among themselves based on individual, potentially biased perspectives. Interaction with the human designer is encouraged by the addition of such notification and exploration facilities as agent protest reports and direct human designer/agent collaboration. These facilities provide the designer with a powerful exploration environment where users are free to engage in private conversations with any combination of Domain, Space, and Monitor agents. The resulting system provides an environment where the human designer forms a partnership with computer-based agents in achieving common design goals and objectives.

CONCLUSION

Although the CAD Research Center has since 1994, by necessity, shifted its focus from the architectural design field to mostly military decision-support applications, our current work is clearly based on experience gained with the ICADS and KOALA prototypes. Apart from advances in object-oriented software engineering concepts and practices, improvements in computer languages, and order of magnitude more powerful hardware, it is in the area of information representation that our recent systems such as IMMACCS (Pohl et al. 1999) have gained most. Today, with the assistance of powerful software development tools we are able to construct more comprehensive and deeper object models to describe the ontology of an application domain. In particular, these tools allow us to more accurately represent a wider range of relationships among objects.

The complexity of the information representation in our collaborative decision-support systems is increasing, while the agents are becoming more numerous and simpler. In ICADS, the underlying frame-based semantic network of objects is relatively sparse in relationships because of the inherent shortcomings of the modeling environment and our lack of understanding of the significance of object associations at that time. Consequently these associations had to be incorporated in the rules of the agents, and this made the tasks of designing and implementing agents unnecessarily complicated. Today, we spend much more time on the design of the ontology and much less time on the design and implementation of agents. Benefits of this trend are readily discernable in terms of: a primary focus on design (up to 50% of the total effort) rather than implementation; the ease with which agents are implemented and modified; the

relatively error-free nature of the systems (i.e., the entire ‘debugging’ process has been reduced from weeks to days); the ability to automatically generate and update system design documentation; the virtual elimination of system design errors; and, the closer collaboration during the design stage with the eventual users of the system.

REFERENCES

- Durrant-Whyte, H. F. (1988). *Integration, Coordination and Control of Multi-Sensor Robot Systems*. Kluwer Academic, Boston, Massachusetts.
- Gero J., M. Maher and W. Zhang (1988); 'Chunking Structural Design Knowledge as Prototypes'; Working Paper, The Architectural Computing Unit, Department of Architectural and Design Science, University of Sydney, Sydney, Australia.
- Hayes-Roth B., M. Vaughan-Johnson, A. Garvey and M. Hewett (1986); ‘Application of the BB1 Blackboard Control Architecture to Arrangement Assembly Tasks’; *The International Journal for Artificial Intelligence in Engineering*, 1(2) (pp.85-94).
- Myers L., J. Pohl, J. Cotton, J. Snyder, K.J. Pohl, S. Chien, S. Aly and T. Rodriguez (1993); "Object Representation and the ICADS-Kernel Design", Technical Report, CADRU-08-93, CAD Research Center, Design and Construction Institute, College of Architecture and Environmental Design, Cal Poly, San Luis Obispo, CA, January.
- NASA (1989); ‘CLIPS Reference Manual (Version 4.3)’; Artificial Intelligence Section, Lyndon B. Johnson Space Center, NASA, May.
- Pan J. and J. Tenenbaum (1991); ‘Toward an Intelligent Agent Framework for Enterprise Integration’; *Proc. Ninth National Conference on Artificial Intelligence*, vol.1, San Diego, California, July 14-19 (pp.206-212).
- Pan, X. (2000); 'An Experimental Adaptive Expert System'; Intersymp-2000, in Pohl, J. and T. Fowler (eds.) *Focus Symposium on Advances in Collaborative Decision-Support Systems*, Baden-Baden, Germany, Aug.
- Pohl, J., Myers, L., Chapman, A. and Cotton, J. (1989); 'ICADS: Working Model Version 1'; Technical Report, CADRU-03-89, CAD Research Unit, Design Institute, California Polytechnic State University, San Luis Obispo, California.
- Pohl J, J. LaPorta, K. Pohl and J. Snyder (1992); "AEDOT Prototype(1.1): An Implementation of the ICADS Model", Technical Report, CADRU-07-92, CAD Research Center, Design

and Construction Institute, College of Architecture and Environmental Design, Cal Poly, San Luis Obispo, CA, August.

Pohl J., L. Myers and A. Chapman (1994); 'Thoughts on the Evolution of Computer-Aided Design'; CAD Research Center, Technical Report (CADRU-09-94), Cal Poly, San Luis Obispo, California (pp.7-34).

Pohl J. (1995); 'The Representation Problem in CAD Systems: Solution Approaches'; InterSymp-95, in Pohl J. (ed.) Focus Symposium on Advances in Cooperative Computer-Assisted Environmental Design Systems, Baden-Baden, Germany, Aug.16.

Pohl J. (1991); 'ICADS: An Intelligent Building Design System'; Proc. European Symposium on Management, Quality and Economics in Housing and Other Buildings, Lisbon, Portugal, Sep.30.

Pohl K. (1996); 'KOALA: An Object-Agent Design System'; InterSymp-96, in Pohl J. (ed.) Focus Symposium on Advances in Cooperative Environmental Design Systems, Baden-Baden, Germany, Aug.14-18 (pp.81-92).

Pohl J. and L. Myers (1997); 'ICADS: An Integrated Knowledge-Based CAD System'; in Tommelein I. (ed.) Expert Systems for Civil Engineers: Integration Issues, American Society of Civil Engineers (ASCE), 345 East 47th Street, New York, NY 10017-2398.

Schön, D.A. (1988); "Designing: Rules, Types and Worlds", Design Studies, 9(3), July.

Simon H. A. (1996); 'The Sciences of the Artificial' 3rd ed., MIT Press, Cambridge, Massachusetts (pp.111-113).