Merging Artificial Intelligence with CAD: Intelligent, Model-Based Engineering

By
Dr. Raymond E. Levitt

TECHNICAL REPORT
Number 28

October, 1990

If you would like to contact the authors please write to:

c/o CIFE, Civil Engineering,
Stanford University,
Terman Engineering Center
Mail Code: 4020
Stanford, CA 95305-4020

Henry M. Shaw Lecture for 1989

Department of Civil Engineering
North Carolina State University

Merging Artificial Intelligence with CAD: Intelligent, Model-Based Engineering

by

Dr. Raymond E. Levitt

Professor of Civil Engineering and
Associate Director, Center for Integrated Facility Engineering
Stanford University
Stanford, California

1. INTRODUCTION

Many physical phenomena of interest to engineers can be described and predicted using the language of continuous mathematics. Until about 1960 engineers relied heavily on their ability to manipulate differential equations in order to comprehend and predict the behavior of all but the most simple engineering systems. Computers began to play an increasingly important part in engineering practice during the 1960s, as computer algorithms that implemented numerical techniques for solving differential equations rapidly displaced closed form mathematical analysis by engineers. Computer-based numerical techniques have now almost completely replaced direct mathematical manipulation for analysis tasks in many fields of engineering.

A second major thrust of computers in engineering has been to provide support for the production of engineering drawings. Early computer-aided drafting (CAD) systems were expensive to buy and difficult to learn, requiring three-shift use and extensive training efforts in many engineering firms. Hence, widespread adoption of the technology was relatively limited before 1980. Computer-aided drafting became ubiquitous during the 1980s as personal computers and their software coevolved in capability and drastically reduced the hardware, software and training cost per drafting seat.

The early CAD systems were two dimensional drafting systems and merely replaced mechanical drawing boards as tools for drawing production, much as the early computer algorithms had replaced mechanical calculators for numerical problem solving. Once CAD systems evolved into three-dimensional modeling tools where design features and components were explicitly recognized, and were provided with links to descriptive textual or numerical data, engineers could use them to record decisions about component selection and layout.

Analysis, drafting and layout are important and challenging engineering tasks and computers have provided powerful productivity gains in all of these areas. However, engineers also spend a great deal of time solving synthesis, diagnosis, planning and other problems which are not readily modeled using numerical algorithms. Humans engaged in synthesis, diagnosis and similar tasks employ pattern matching, deduction, and related cognitive processes which involve manipulating non-numerical symbols and concepts, and which are not easily modeled using the procedural computer languages such as FORTRAN that were developed to automate numerical computations.

Tools like the abacus, the slide-rule, and the mechanical calculator have long been used as aids in solving numerical problems, and early computers were perceived by many as merely electronic extensions of such tools. However, neither the abacus nor the first generation of computers was able to duplicate human problem solving capabilities for non-numerical tasks—the latter were widely believed to require human intelligence. As a result, the field of computing which aimed to model and replicate human problem solving for such tasks became known—to the chagrin of some of its proponents—as the field of *Artificial Intelligence* (AI). Symbolic processing "fifth generation" computer software languages and tools were specifically developed by AI scientists to facilitate modeling the non-numerical cognitive processes of pattern matching, search and deduction used in tasks such as synthesis, diagnosis, planning, and interpretation.

In about the mid 1980s, AI techniques—in particular knowledge-based expert systems (KBES)—began to be used by engineers to support diagnosis, selection and monitoring tasks on a routine basis. Digital Equipment Corporation's *R-1* system for configuring VAX minicomputers is probably the best known KBES application of this era [Barker 1989] and is claimed to have saved DEC millions of dollars annually [Feigenbaum 1988]. Literally thousands of other small, focused AI applications, many running on personal computers, were introduced by engineers in both private and public engineering organizations of all sizes in the second half of the 1980s.

Most of the KBES applications introduced during this time period were consultation-style expert systems modeled after MYCIN (an early medical diagnosis system) [Buchanan 1984] or PROSPECTOR (a pioneering mineral prospecting system) [Campbell 1982]. In such systems, the user provides the data needed by the program in response to a series of questions posed by the system in a data-driven order. While this was adequate for small scope, stand-alone problems, KBES to support design tasks need to be tightly integrated with CAD and analysis applications to free human users from an overwhelming data entry load.

In particular, as engineers begin to design at CAD workstations rather than on paper, new opportunities arise for automated data sharing and coordination of specialists' decision-making in real time. The integration of KBES techniques with traditional engineering tools for CAD and analysis is a prerequisite for automating and integrating the numerical and non-numerical engineering tasks associated with planning, designing, manufacturing and operating semiconductors, automobiles, aircraft, buildings, process plants and other engineered systems. This paper lays out a methodology for using AI techniques to create the next generation of decision support systems for engineering design synthesis.

2. A VIEW OF THE DESIGN PROCESS FOR CONSTRUCTED FACILITIES

The design process is often viewed as consisting of three tasks, performed iteratively in increasing detail:

- ❑ A designer first synthesizes one or more candidate solutions aimed at meeting a set of design specifications;
- ❑ The designer then analyses each candidate solution to determine its behavior in terms of important dimensions of the system's performance; and
- ❑ The designer evaluates the performance of each candidate solution to determine whether, and how well, it meets the design specifications. If a solution is deemed unsatisfactory, the dimensions of performance needing improvement are used to guide the next round of synthesis for an improved solution, and so on.

In the case of constructed facilities, computers have principally been used thus far to automate the analysis step of design. Here numerical approximations of differential equations have provided powerful and useful models of the physical behavior of complex structural, mechanical and other building systems. Synthesis, however, has been performed by human designers. CAD or other graphical data entry software tools have supported synthesis primarily by recording and communicating the designers candidate solutions to analysis programs or to other engineers. We will show in this paper how evolving AI-based tools can support design synthesis directly, to the point of automating it completely in many cases. But first we will look at the synthesis component of design in more detail.

Design synthesis for buildings, process plants and many other kinds of constructed facilities can be viewed as consisting of three steps: *select, connect* and *position.* We describe each of them briefly here.

- **Select.** Facilities such as buildings and process plants are increasingly assembled from standard or almost standard components. A major task in design synthesis is to select components, at different levels of grain-size, from libraries of standard components or subsystems. Knowledge about how to select appropriate components, given the operating requirements of the facility and choices already made about interdependent subsystems or components, is one of the skills that experienced designers possess.

- **Connect.** Components must be connected together correctly to perform their function. Properly connected structural members create load paths; piping elements create fluid flow paths; and so on. Development of the topology of components in a design synthesis is often supported by two- or three-dimensional schematics such as P&IDs for process plants, circuit diagrams for electrical systems, and wire-frame models for structural systems.

- **Position.** For buildings, layout of architectural components drives the entire design process. For other kinds of constructed facilities such as process plants, however, layout is performed as a final step, after selecting and connecting components. In either case, experienced designers use experience to guide them in positioning components relative to one another.

In the following section, we will show how AI techniques of rules, frames, and object-oriented programming can be integrated with CAD to support each of these tasks for the synthesis of products composed of standard components.

3. INTEGRATING AI WITH CAD AND ANALYSIS

A great deal of thinking and experimentation has been devoted to the use of AI techniques to augment the capabilities of CAD and analysis tools already used by engineers. The outline of a workable methodology for incorporating AI techniques in engineering computing environments can be abstracted from experience to date. The approach described below is proposed as a structured way to integrate non-numeric reasoning with numeric and graphical modeling tools for diagnosis, synthesis and planning tasks in engineering. Disciplined experimentation with and refinement of a methodology for building integrated engineering software tools is now both possible and desirable. The alternative is to continue with ad-hoc application development, repeating each other's mistakes as well as successes.

The methodology builds on existing CAD and analysis tools and practice and derives data on-line from applications wherever possible. We begin by listing the kinds of data that can be obtained from CAD databases and schematics, and discuss some modest enhancements of CAD systems to integrate topological data about the components of a constructed facility with data describing their geometry, materials properties and administrative attributes.

3.1. OBTAINING COMPONENT DESCRIPTIONS FROM CAD DATABASES

As discussed above, selection, refinement and topological mapping of system components are the principal tasks of design synthesis in many engineering disciplines. In particular, CAD systems have become extremely useful not just for drafting but also for the synthesis tasks of component selection and topological mapping. This transition from Computer-Aided Drafting to Computer-Aided Design could occur once CAD systems evolved to a point where they could represent physical features and components as their primitives rather than graphical elements such as points, lines and arcs. As a result of this advance in CAD system capability, which occurred in the mid-1980s, engineers can productively design products ranging from semiconductors and cameras to space shuttles and power plants by synthesizing engineering designs on workstation screens rather than on paper.

Furthermore, for all but the most unique products, designers at CAD workstations can now extract components from a company, vendor or industry database of standard components stored in an appropriate CAD format, and insert them into a particular synthesis. A series of components that has been selected, resized or adapted in some other way, and then appropriately connected to other components in the model, defines a unique *instance* of a product.

For products designed in this way the component records or "blocks" in the CAD database contain—or point to records in an external database which contain—several kinds of descriptive information that can be used by a KBES to support design, manufacturing and operation of the product:

- ❑ **Component geometry** is described in sufficient detail to draw components at the scale used in the CAD system in which the design is being performed. Geometry can also be used to compute attributes such as the surface area, volume, mass, center of mass, radius of gyration or moment of inertia.

- ❑ **Topological information** about the components of a product is provided only implicitly in purely graphical CAD systems by the user at the time that components are situated in a model. Because it is difficult to interpret topology from geometry alone, two-dimensional schematics such as circuit diagrams or piping and instrumentation diagrams (P&IDs) are widely used by engineers to represent device topology. However, schematic diagrams like these have had no computer usable connection with CAD geometric models to date. For complex 3-dimensional products, scale models or full scale mock-ups have often been used to help engineers evaluate the manufacturing or operating concerns associated with geometry and topology. Spatial reasoning techniques can interpret a purely geometrical model to infer topology for simple cases like office building structural frames, but tend to bog down when required to interpret more irregular topologies.

  Consequently, high-level designer interfaces have recently been developed for a number of CAD systems to facilitate the capture of explicit topological data among product components as they are inserted into a 3-D CAD design model. The explicit topological information thus stored in the CAD database can then be used to support several kinds of reasoning using standard KBES techniques [Ito 1989].

- ❑ **Physical properties** of the materials of which the component is made for use by structural, heat flow, chemical or other analysis programs.

- ❑ **Technical specifications** for manufacturing, assembling, testing or operating the components of the engineered system.

- ❑ **Administrative information** needed in the manufacturing process, e.g., vendor name, contract number, payment provisions, etc.

- ❑ **Analysis information** e.g., member or connection forces and moments computed by a structural analysis application, may be needed for diagnostic or design reasoning. Translation packages exist to share geometric data and analysis results between many standard CAD and structural analysis packages.

Several AI programming tools now run concurrently on the same workstations that are used for CAD, or can easily access CAD data over networks. Thus, they can readily extract these types of product information from a CAD database (or an attached relational database) and can automatically create a frame representation of the product that incorporates these data.

Typically, each component is represented in a single frame at whatever level of detail the engineer wishes to reason about the engineered system. A single component frame can thus be used to represent a component which is really a subassembly consisting of hundreds of separate parts, or an individual keyway, hole, or washer in a product. The frames in such a product model can represent geometrical, physical and administrative attributes of a product's components together with the topological structure of the components. All of this information about the structure of a product and the local values of its component attributes is then available in a representation easily accessible to KBESs for several kinds of engineering reasoning.

Up to this point, we have not really enriched the data available in the CAD system; we have just reorganized it into frames. However, frames are more elaborate data structures than database records—they can store knowledge about the behavior of a component along with data about its current state [Fikes 1985]. The following sections show how we can exploit the power of production rules and inheritance to enrich the data in the frames, transforming a database of component data into a knowledge base describing both the state and the behavior of an engineered product.

3.2. HOW COMPONENTS CAN INHERIT BEHAVIOR

The specific components which comprise an engineering system can be viewed as instances of more general classes of components. Some component attributes and their values—especially those describing component behavior—can be defined at the most generic class level to which components belong, e.g., in a class called "building components." Additional attributes and their values representing descriptive properties or behavior of more specific subclasses of components, e.g., "HVAC system components" or "electrical system components," can then be added at each level of specificity and inherited down the abstraction hierarchy to particular instances of each type of component.

The key to inheriting component behavior is that objects retrieved from the CAD system must be correctly associated with the component abstraction hierarchy, i.e., they must automatically be made instances of the correct subclasses. The importance of strict naming conventions for components becomes obvious at this point. For example, with careful enforcement of naming conventions, a rule can be used to find all components labeled BEAM_xxx in the CAD system database, and then attach them automatically as instances of the subclass BEAMS in the frame hierarchy, whence they can inherit beam bending formulae, deflection limits, etc. (Languages like Lisp handle strings effortlessly, so that simple methods or rules to identify and attach instances to an abstraction hierarchy are easy to program.)

Once a component has been correctly placed in its primary class, e.g., BEAM_263 has been made an instance of BEAMS, it is easy to generate other links that can provide more specific behavior using additional rules or procedures which reason about attributes of the component, e.g., its material type. The following rule shows how easy this is to do:

```
((If      (?BEAM is in class BEAMS)

And       (the MATERIAL_TYPE of ?BEAM is CONCRETE))

(Then     (?BEAM is in class CONCRETE_COMPONENTS)))
```

This rule would search over all instances of BEAMS and would add the subclass parent CONCRETE_COMPONENTS to any instances of the class BEAMS whose material type was concrete, causing them to inherit additional behavior from the subclass CONCRETE_COMPONENTS. Thus, by firing a set of rules of this type on the frames generated from a CAD model of a product, we are able automatically to add knowledge about component behavior to a knowledge base that previously contained only information about component geometry, topology and material type.

3.3. HOW COMPONENTS CAN DEDUCE THEIR FUNCTION

A KBES can deduce a substantial amount of knowledge about the roles that individual components such as beams, valves, shafts or switches play in the functioning of a product by reasoning about the part-of hierarchy for the product and the topological links among its components. Thus, by noting that a beam is *part-of* a liquid nitrogen supply subsystem and is *connected-to* the liquid nitrogen pump platform, we might conclude that its role or function in the product is to provide structural support for the pump platform. This allows the beam to reference the weight of the pump in computing its size.

Similarly, an electrical conductor that was *part-of* the liquid Nitrogen supply system, and was *upstream of* the pump (in terms of electrical circuit topology) could reference the pump's voltage and starting horsepower to determine its own required current carrying capacity, and so on.

Knowledge of component function provides significant leverage in performing tasks like design, simulation or diagnosis. However, strict naming conventions are extremely important here, too, to ensure that components are properly attached to the part-of and abstraction hierarchies to which they should belong so that component function can be correctly deduced.

The RATAS building product model developed in Finland was one of the first systems to exploit this capability in the building domain [Bjork 1988]. We will illustrate the use of linked abstraction hierarchies and part-of hierarchies to infer component behavior and function in describing the Intelligent Boiler Design System [Riitahuhta 1988] later in the paper.

3.4. GENERATING SYSTEM BEHAVIOR VIA MODEL-BASED REASONING

Model-Based Reasoning (MBR) is a style of KBES modeling that can be used to predict the behavior of the modeled system by qualitative or quantitative simulation [Kunz 1989]. Inherited methods give components the correct behavior in the context of local descriptive data. The topological information stored with each component allows such a model to deal with interactions among its components, e.g., current flow between connected components of an electronic circuit, fluid flow through a piping system, load paths through a structure, or the propagation of vibration through an airframe.

For tasks such as product configuration and diagnosis, qualitative modeling techniques are often perfectly adequate for simulation while at the same time offering the advantage of being able to explain their conclusions. However, qualitative simulation will not always produce definitive results; often, two or more opposing qualitative effects must be quantified in order to determine which dominates. Thus, we can might develop KBESs which move through a natural progression from purely qualitative simulation through simple quantitative simulation, e.g., a "back-of-the-envelope" approximation, to detailed numerical modeling. This corresponds closely to the way experienced engineers work. We use detailed numerical analysis in very focused ways to resolve specific design questions whose answers may not be clear from qualitative or simple quantitative analysis.

We summarize the four basic points of the model-based reasoning approach to design synthesis for products consisting of standard components or features as follows:

1. Components needed for a product are retrieved from a library of standard components and attached to their primary parents in a component abstraction hierarchy based upon standard naming conventions. Rules or procedures that reference additional attributes of named components then generate multiple abstraction links for components. This provides components with knowledge about arbitrarily complex behaviors by inheritance from multiple subclass parents.

2. Product structure is represented by a part-of hierarchy for its components and by topological links—either explicitly provided by the product designer in

schematic or CAD form as components are added to a design in the synthesis stage, or automatically deduced by spatial reasoning techniques.

3. Component function is deduced by reasoning about links in a generic product part-of hierarchy and/or by reasoning about component topological relationships.

4. Product behavior is deduced by simulation—qualitative, quantitative or both—of the product, using inherited component behaviors and local attribute values to generate component behavior. Again, topological links capture interactions among components and subsystems.

This methodology is summarized graphically in Figure 1.

****
INSERT FIGURE 1 ABOUT HERE
****

One of the principal advantages of this approach to design synthesis is the extensive use of generic component libraries represented as frames, whose attributes and behavior can be inherited by instances of the components in engineered products or systems. This provides two important advantages.

- ❑ First, it drastically reduces the number of new rules that need to be employed in a given MBR engineering application. As a company develops new products which incorporate generic components, much of the system behavior is already captured in the inherited behavior of the components. The volume of new rules to be encoded in order to define synthesis knowledge for new products will thus increase less than linearly—rather than exponentially as would be the case with a rule-based system—as long as new products share some of the same components as previously modeled products.

- ❑ Second, frame-based inheritance is a much easier form of deduction than rule-based inference to program and maintain. Thus, MBR systems implemented in this way are far easier for an organization to develop and maintain than corporate knowledge bases consisting only of production rules such as are used for the R-1 application by DEC [Riitahuhta 1988].

3.5. ENGINEERING APPLICATIONS OF MBR

We have described a methodology whereby model-based reasoning can be implemented using KBES and CAD systems together to transform a database of information about a product's components into a knowledge base of component properties and behavior, and then perform design synthesis with this rich knowledge base.

This type of approach has been applied in the LSC Advisor for architectural code checking [Dym 1988], the SightPlan system for construction site layout [Levitt 1989] and the IBDS system for automating boiler design [Riitahuhta 1988]. We describe the IBDS system in some detail next.

4. AN EXAMPLE OF AI-CAD INTEGRATION: THE INTELLIGENT BOILER DESIGN SYSTEM OF TAMPELLA

AI techniques for design synthesis are just now entering into commercial application. In this section we provide a brief case study to show how one company is using a knowledge-based design automation system to automate the preliminary design of power generation boilers. This case study demonstrates the integrated and operational use of state-of-the-art AI-CAD integration techniques in a real engineering environment.

4.1. BACKGROUND OF IBDS

Tampella Power Industries of Tampere, Finland, designs and manufactures boilers and other components of power plants. Since April 1987 the company has been using Design++, a high-level design automation AI language that follows the model-based reasoning (MBR) paradigm described above for boiler design. The boiler design expert system developed using Design++ is integrated with a ComputerVision CAD system, a VAX-based engineering analysis system, and an Oracle relational database system [Riitahuhta 1988]. One part of this application conducted by Tampella addressed the design of upper circulation piping (Figure 2).

****

INSERT FIGURE 2 ABOUT HERE

****

Here, the upper circulation pipes refer to the water-collecting pipes of the boiler furnace walls which return the boiler water into the drum.

Modeling for the IBDS pipe layout system was divided into three stages.

- Defining the positions of the drum connections,
- Defining the positions of the connections of the wall collecting headers, and

- Defining the routing of the pipelines from the drum connections to the corresponding connections of the wall collecting headers.

In principle, the layout of the drum connections is determined by standard construction, but the number and positions of the connections have to be determined separately for each project. The positions of the connections for a given boiler are dictated by the need to avoid interference with the downcomers. There are construction standards for the layout of the wall connections. Solutions for the particular project are provided again by the specific obstructions resulting from different boiler sizes. Tampella has not been able to standardize these structures; rather, the structure is determined for each project using design rules.

There are several obstructions, including the upper headers, the weld joints, and the lifting lugs of the header needed for hanging both the header and the furnace wall it supports from the upper support structures. Boiler downcomers, boiler suspension rods, parallel circulation pipes, and other piping systems are additional obstructions. With the aid of design rules implemented in Design++, the positions of the drum connections and wall collecting header connections are determined and the boiler pipes are positioned automatically, one at a time, to avoid all of these obstacles .

4.2. ARCHITECTURE OF IBDS

The Design++ framework is implemented as a knowledge base in IntelliCorp's KEE, which runs under Common Lisp. IBDS is made up of a series of linked knowledge bases implemented on top of Design++. The original version of the IBDS was implemented on a Symbolics AI workstation and interfaced to other computers. The current version of Design++ runs in a UNIX environment on Sun workstations. This allows applications like IBDS to interface and share data with relational databases, document preparation software, engineering analysis programs, CAD programs, and other applications in the UNIX environment.

4.3. REPRESENTATION IN IBDS

Tampella first developed a set of generic component libraries stored as frames within Design++ (as Design++ is built on top of KEE, these libraries are stored as standard KEE frames but are accessed through an engineering design user interface). Design++ uses a form of **prettyprinted** text file to represent product part-of hierarchies. Tampella therefore made up one of these part-of hierarchies (which can recursively reference more detailed part-of hierarchies) for each type of power boiler that it wished to design. Configuration knowledge was represented in rules defined for classes of assemblies and classes of parts in generic component libraries.

4.4. REASONING IN IBDS

Attribute values are propagated via demons. Rules fire as components are defined to propagate constraints from the attributes of one component to the relevant attributes of others. For instance, as a pump is selected, its power consumption is propagated to conductors, transformers, circuit breakers, or other electrical components that need this information to select or size themselves; its weight is propagated to structural components that provide it support; its output diameter is propagated to the pipe and flange that connect to its output side; and so on. This is classic object-oriented programming (OOP) as described in Chapter 6.

Design++'s high-level design rule language facilitates the implementation of spatial reasoning (e.g., about clearances) and some kinds of electro-mechanical reasoning in terse rules, but any desired relationship among component attributes can be defined in Lisp as part of such demons. Substantial Lisp methods are used to carry out some of the more complex types of geometric reasoning in IBDS (e.g., routing of pipes).

4.5. EXPLANATION IN IBDS

The current version of Design++ uses AutoCad as an internal visualization tool. This permits the user to see the evolving design in three dimensions, colored by layers or subsystems, as desired. The design visualization can be rotated, panned, zoomed, and so on, from within AutoCad, so the user can inspect the results of design decisions made to date. The user can backtrack to make changes, and all affected components are automatically redesigned. This permits consideration of multiple alternatives, each of which is guaranteed to be consistent with all design rules in the system, in a fraction of the time that would be required to do this manually or with conventional CAD systems.

Design++ has the ability to represent reports as assemblies of components (paragraphs, subsections, and so on.). Also, there are interfaces between Design++ and document processing languages such as InterLeaf. These two features allow the automatic generation of reports, such as bills of quantities, specifications, warranties, procedure manuals, and so on, based upon the design model. AutoCad images from the design model can be pasted into such reports. The AutoCad model can be used to produce preliminary or working drawings or to generate CAD files in the format of other three dimensional CAD systems via its DXF interchange format. Finally, the model has been used by Tampella to generate manufacturing instructions in numerical control (NC) format for robotic pipe benders or other automated manufacturing equipment.

4.6. CURRENT STATUS OF IBDS

The power of the IBDS system is based on the large amount of knowledge of the domain that can be captured in the design rule language, project structure model, and component libraries in Design++. When the amount of knowledge increases, knowledge management becomes a significant problem. In this system, the knowledge management problem has been solved by utilizing an object-oriented approach to component descriptions and by exploiting relational databases.

The benefit obtained through automation in the design of the upper circulation pipes is summarized by Tampella's engineering staff [Riitahuhta 1988] as follows (emphasis added):

- Using manual design, the design work took two *months*.
- With ComputerVision´s conventional plant design software, design took two *weeks*.
- Using the IBDS expert system together with the ComputerVision CADD system, a design can be completed in two *days*.

Thus, AI techniques have been exploited to automate many aspects of routine design for a semi-custom product. The techniques used in this example parallel very closely the MBR approach. There currently exist a number of off-the-shelf design automation tools like Design++ (including ICAD and Concept Modeler) that can facilitate the development of applications like this. The best of these design automation tools not only provide rules, frames and OOP for representation and reasoning; they also serve as the "glue" to integrate CAD, analysis, database, document preparation, and manufacturing automation software tools.

5. CONCLUSIONS

This paper has argued that model-based reasoning provides the beginnings of a structured methodology for building knowledge-based engineering systems to support design synthesis. We have explained and demonstrated how the AI techniques of production rules, frames and OOP can be integrated with traditional engineering CAD, database and analysis software to perform model-based reasoning in ways that leverage the advantages of each.

An example application was described to show that AI and CAD can be usefully merged in the style proposed in this paper to automate design synthesis for semi-custom products such as industrial boilers. The author is aware of completed or ongoing work using this style of AI-CAD integration for automating the synthesis of building air-conditioning systems and elevators, as well as of complete manufacturing plants for paper and wire cables.

Synthesis of non-routine designs is a far more difficult task in which humans may use analogy, mutation and other less-well understood mental processes for generating candidate solutions that might satisfy unusual specifications. Work in this area is still in the early research stages and commercially interesting results are probably several years away [Gero 1987]. However, where facilities are assembled from relatively standard components in product structures that contain many of the same high-level subsystems, current technology for merging AI with CAD can provide order of magnitude increases in engineering productivity with improved preliminary designs and cost estimates, and can be used to integrate analysis and manufacturing software tools at all stages of the facility engineering life cycle.

6. ACKNOWLEDGEMENTS

Many of the ideas presented in the paper arose out of seminar discussions with an unusually talented and articulate group of students, faculty and industry colleagues in CIFE. At the risk of offending anyone I might forget, I would like to acknowledge the contributions to my thinking made by a series of discussions and debates on this subject with Alan Axworthy, Paul Chinowsky, Lai Chua, Tony Confrey, Adnan Darwiche, Hossam El-Bibany, Cristina Bicharra, Ed Feigenbaum, Martin Fischer, Thomas Froese, Renate Fruchter, Gabe Gross, Barbara Hayes-Roth, Craig Howard, Kenji Ito, Deepak Jain, Nabil Kartam, Matti Katajamäki, Tapio Karras, John Kunz, Kincho Law, Hannu Lehtimäki, Douglas Phan, Asko Riitahuhta, Iris Tommelein, Yasu Ueno, and Lloyd Waugh.

The SightPlan research referenced in this paper was supported by the *National Science Foundation Grant # MSM-86-13126.* Additional support for research on model-based planning and scheduling has been provided by *Stanford's Center for Integrated Facility Engineering.* This support is gratefully acknowledged.

7. REFERENCES

[Barker 1989] V. E. Barker and D. E. O'Connor, "Expert Systems for Configuration at Digital: XCON and Beyond," *Communications of the ACM* 32 (3), 1989.

[Bjork 1987] B-C. Bjork, *RATAS: A Proposed Finnish Building Product Model*, Studies in Environmental Research No. T6, Helsinki University of Technology, Otaniemi, Finland, 1987.

[Buchanan 1984]	B. G. Buchanan and E. H. Shortliffe, *Rule-Based Expert Systems*, Addison-Wesley, Reading, MA, 1984.
[Dym 1988]	C. L. Dym, R. P. Henchey, E. A. Delis, and S. Gonick, "Representation and Control Issues in Automated Architectural Code Checking," *Computer-Aided Design* 20 (3), 1988.
[Feigenbaum 1988]	E. A. Feigenbaum, P. McCorduck, and H. P. Nii, *The Rise of the Expert Company*, Times Books, New York, 1988.
[Fikes 1985]	R. Fikes and T. Kehler, "The Role of Frame-Based Representation in Reasoning," *Communications of the ACM* 28 (9), 1985.
[Gero 1987]	J. Gero, "Prototypes: A New Schema for Knowledge-Based Design," working paper, Architectural Computing Unit, University of Sydney, Australia, 1987.
[Ito 1989]	K. Ito, U. Yasumasa, R. E. Levitt, and A. Darwiche, *Linking Knowledge-Based Systems to CAD Design Data with an Object-Oriented Building Product Model*, Working Paper No. 7, Center for Integrated Facility Engineering, Stanford University, Stanford, CA, 1989.
[Kunz 1989]	J. C. Kunz, M. J. Stelzner, and M. D. Williams, "From Classic Expert Systems to Models: Introduction to a Methodology for Building Model-Based Systems," in G. Guida and C. Tasso (Editors), *Topics in Expert System Design*, North-Holland, Amsterdam, 1989.
[Levitt 1989]	R. E. Levitt, I. D. Tommelein, B. Hayes-Roth, and T. Confrey, *SightPlan: A Blackboard Expert System for Constraint Based Spatial Reasoning About Construction Site Layout*, Technical Report No. 020, Center for Integrated Facility Engineering, Stanford University, Stanford, CA, 1990.
[Riitahuhta 1988]	A. Riitahuhta, "Systematic Engineering Design and Use of an Expert System in Boiler Plant Design," *Proceedings of the ICED International Conference on Engineering Design*, Budapest, Hungary, 1988.

8. FIGURES

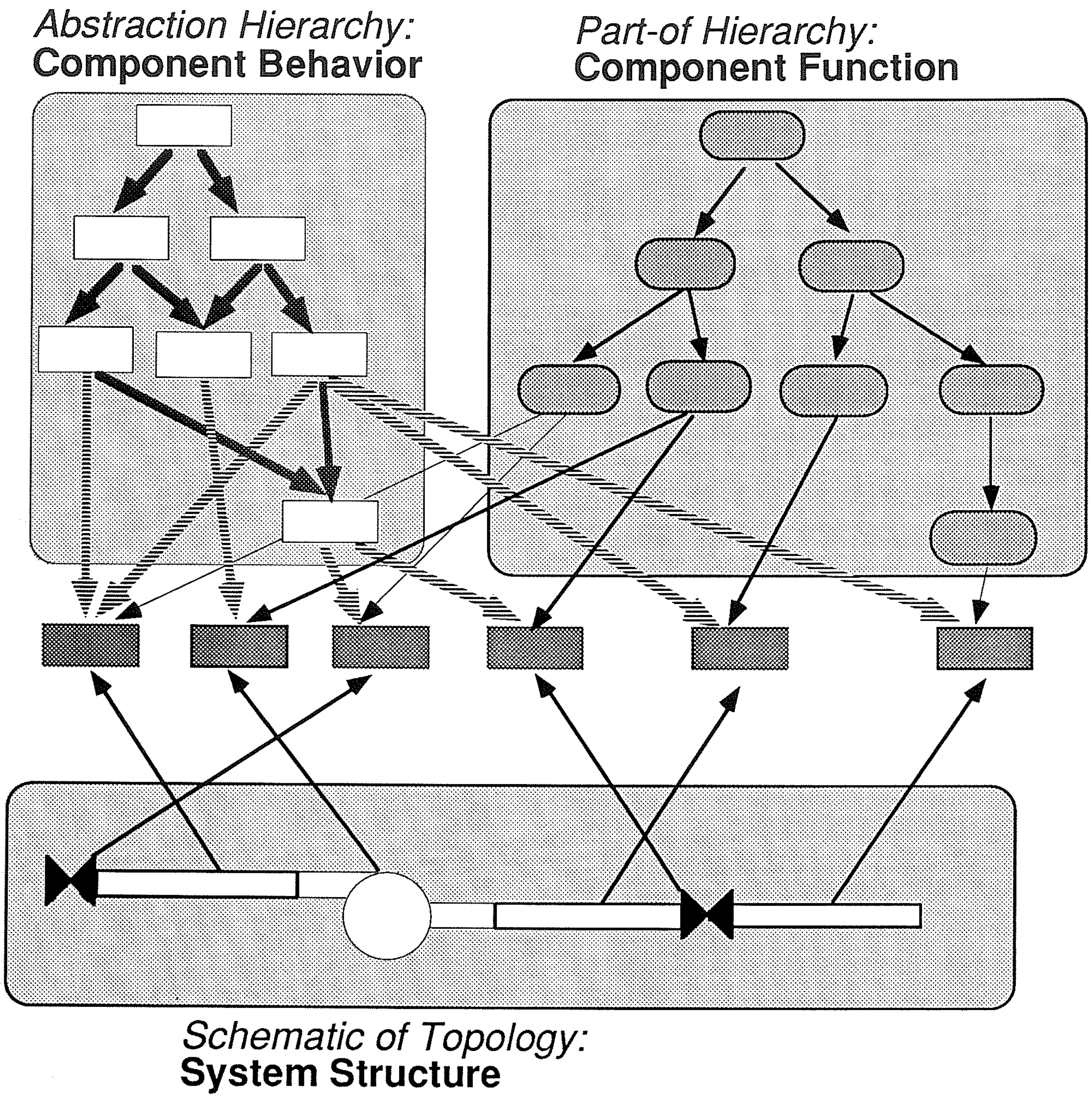

Figure 1: The elements of AI-CAD Integration. Components (shaded boxes) are inserted into a design knowledge base from CAD or database component libraries and connected to appropriate parent subclasses in a frame hierarchy (white boxes) from which they inherit appropriate behavior. Component function is derived from their position in the part-of hierarchy (shaded ovals) and the schematic diagram (shown at the bottom of the figure).

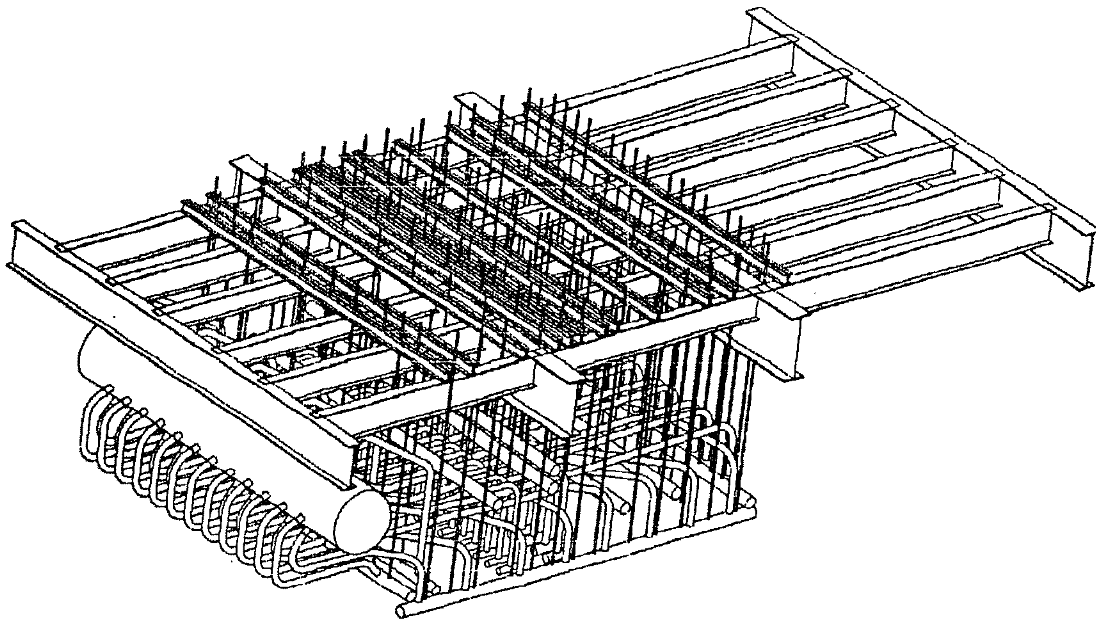

Figure 2 The upper circulation pipework for a power plant boiler (after [Riitahuhta 1988]). Design++, a high-level AI design automation language, was used to generate the component layout for this boiler using the MBR approach described in this paper.