ARCHITECTURE OF AN INTEGRATED KNOWLEDGE BASED ENVIRONMENT FOR STRUCTURAL ENGINEERING APPLICATIONS

by

D.R. Rehak, H.C. Howard and D. Sriram

DRC-12-19-84
December, 1984

Architecture of an Integrated Knowledge Based Environment for Structural Engineering Applications[1]

Daniel R. Rehak,[2] H. Craig Howard[3] and Duvvuru Sriram[4]

Department of Civil Engineering
Carnegie-Mellon University
Pittsburgh, PA. 15213

Abstract

Structural engineering CAD systems are evolving to incorporate knowledge based processing techniques for design applications, standards processing and database access and interfacing. A number of technical problems which have limited the development of integrated CAD systems are discussed, and the role of knowledge based programs in addressing these problems is presented. The conceptual architecture of a system environment for use in building integrated structural engineering CAD systems is presented. This architecture is based on a distributed network of cooperating, knowledge based processing components. Research in progress aimed at building such a system is also presented.

1. Introduction

1.1. Overview

Structural engineering computer applications are evolving to incorporate knowledge based processing techniques to address the ill-structured nature of the structural engineering process.[5] Work is ongoing to coordinate and integrate a number of structural engineering computer applications, developing the framework for an integrated structural engineering CAD (Computer-Aided Design) system. Based on the needs and the nature of the structural engineering process, a *conceptual* architecture of an integrated knowledge based CAD environment for structural engineering applications has been developed. This conceptual architecture is heavily influenced by the nature of the structural engineering process. It defines the character and interactions of the basic components, and it is formulated to provide the framework or environment for building actual knowledge based application systems which will incorporate more detailed structural engineering processes.

[1] Printed in *Knowledge Engineering in Computer-Aided Design,* J. Gero, ed., IFIP Working Group 5.2 Conference, Budapest, Hungary, September, North-Holland, 1984.

[2] Assistant Professor

[3] Graduate Research Assistant

[4] Graduate Research Assistant

[5] Structural engineering is used in its broadest sense. It is not limited to the design and analysis of the load carrying mechanism of a structure. Herein it implies all aspects and disciplines involved in the design and construction of a complete structure such as a building or a bridge.

The remainder of this paper describes the motivation, issues and design of the system with an emphasis on the capabilities provided and techniques used. The emphasis of the discussion is on the conceptual nature of the system and the problems which must be addressed; detailed examples of components and problems are beyond the scope of the presentation.

1.2. Motivation

The first CAD applications were primarily drafting and graphics systems. Analysis programs, such as finite element analysis, also have played an important role in the application of computers to engineering. Excellent progress has been made in developing high quality programs with sophisticated capabilities in disciplines such as geometric modeling, finite element analysis, graphics display and drafting, circuit routing, circuit simulation, numerical control, etc. Many CAD systems have evolved from such programs as vertically integrated sets of software combining both analysis and graphical capabilities for a single engineering problem solving task.

In a move to integrate computer use throughout the design process, a horizontal integration of applications across domains is also underway. This integration addresses the information flow and communications problems which are a major aspect of the inter-disciplinary design process. Centralized database management provides the basis for linking various design applications into an integrated system. Thus CAD systems have evolved from first generation drafting and analysis programs to second generation systems which provide integrated design environments based on centralized database management. The architecture of such a system for mechanical CAD/CAM is shown in Figure 1.

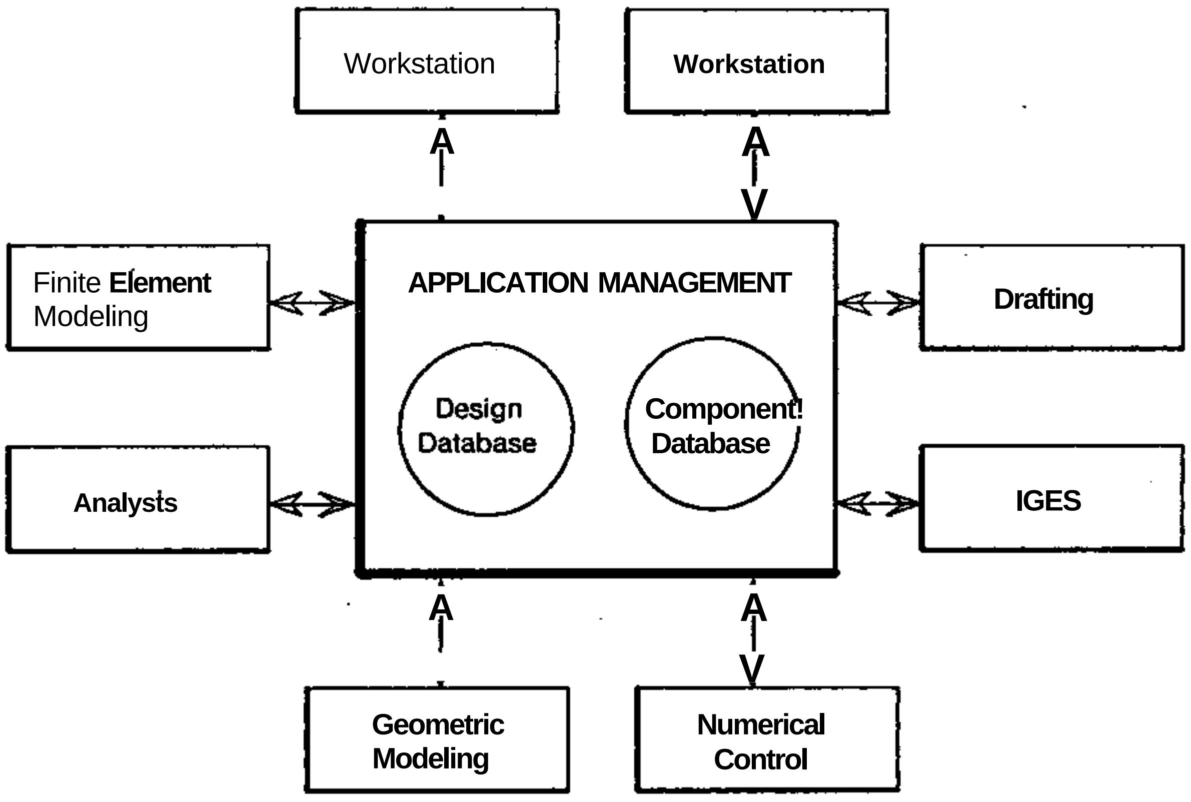


Figure 1: Second Generation CAD/CAM System Structure (after [Christman 83])

Today's CAD systems have extensive capabilities, but are currently limited to problem solving tasks which are algorithmic or deterministic in nature. However, many engineering design activities, including synthesis, evaluation, modeling and the *creative* aspects of the design processes, are ill-structured and require heuristic solutions based on judgment and experience. Due to their nondeterministic nature, these problem solving tasks have not been successfully computerized. Knowledge based programming techniques enable the computer to assist in such ill-structured problem solving tasks. Since such tasks are a major aspect of the engineering design process, it is expected that the next generation of CAD system will evolve to incorporate knowledge based processing [Preiss 83].

Work on producing an integrated design environment for structural engineering has been underway for two decades. Previous efforts include systems such as *ICES* [Roos 66], *GENESYS* [Genesys 76] atid *POLO* [Lopez 72]. In the late seventies it was realized that the current state of computer technology, as applied to engineering problems, was insufficient to address the needs and requirements of the desired structural engineering CAD systems. The emerging technologies of knowledge based expert systems and relational database management were identified as being useful in addressing the problems that had stymied development of such applications. A conceptual architecture for an integrated civil engineering design environment *(CAESE,* Computer Aided Engineering Software Environment) based on these technologies was developed [Rehak 81]. Over the past three years, work has been progressing on extending and revising the initial design and on building prototype components of such a system.

This article updates the previous work and presents the authors' current thinking on issues and the architecture of an integrated CAD system for structural engineering. Since there are some unique aspects to the structural engineering process, a short discussion of the nature of the process is presented. This is followed by a brief summary of knowledge based applications in the structural engineering domain. From this background, a set of motivating issues and design concepts which lead to the conceptual architecture of a knowledge based integrated CAD environment for structural engineering is developed and discussed. The presentation of the system architecture includes a discussion of data management, the role of distributed personal computing, and thoughts on software approaches.

It must be emphasized that the CAD system architecture presented in this paper is under constant revision. As additional prototype components are built, and as the basic underlying hardware and software technologies evolve, new issues appear, and there is an ongoing adjustment of direction. The discussion is based on current thinking and experiences over the past few years and expresses the long term goals and **the short** term directions towards meeting those goals.

2. Issues in Structural Engineering Computer Applications

Due to the dispersed nature of the construction industry, the influence of the regulatory process on civil engineering design and construction, and the unique, one-of-a-kind nature of projects, structural engineering computer applications must deal with some unique issues which do not affect mechanical engineering or electronic CAD applications. Any integrated CAD design application in the civil or structural engineering domain must address these realities, and thus will contain special features. But within these constraints, the basic heuristic and iterative nature of the structural design process is similar to the design process in other disciplines.

2.1. The Structural Engineering Process

The structural engineering process begins with a client's need for a structure co provide some function. The final product is a detailed set of specifications describing a structure capable of meeting those needs. This is followed by the actual construction of the structure. As with other design processes, structural engineering is a multi-step, iterative process. The major steps of the process are:

- *Project Planning.* The project planning step (also denoted predesign or programming) takes the basic set of requirements and needs for a structure and expands these to produce a formal set of requirements and design criteria which are the basis for all design work done in latter stages.

- *Conceptual Synthesis.* The conceptual synthesis step (also denoted conceptual design or preliminary design) takes the project requirements and develops a number of alternative designs to some level of detail (typically designing key elements and completing about one third of the total design). This synthesis step consists of a number of different, interrelated processes, all of which interact to produce the preliminary design of the structure. These processes include:

    - *Layouts and Schematics Generation* — Develop and evaluate a number of alternatives for the basic layout of the structure, the configuration of the structural load-carrying system and the configuration of other components (constructed elements) of the structure.

    - *Cost Analysis* — Estimate the construction cost and operating (life-cycle) costs of the design alternatives.

    - *Functional Planning* — Analyze and evaluate the functional capabilities of the design to meet the required design criteria as set forth in the project plan or by regulatory bodies. An example is the compliance checking of life-safety and access requirements for a building.

    - *Constructed Element Design* — Synthesize, analyze, and evaluate each of the subsystems which comprise the complete structure to insure they are technically competent. Some typical analysis tasks for a building include structural analysis (evaluation of the load-carrying capacity of the structure), substructure analysis, electrical distribution system analysis, mechanical system analysis, HVAC system analysis, etc. A similar set of synthesis and evaluation tasks exist for each subsystem, and analogies exist for other classes of structures.

    - *Preliminary Design Documents Production* — Generate a set of documents describing the selected design alternative. Documents include schematic drawings of the structure and outline design specifications.

- *Detailed Synthesis.* The detailed synthesis (or detailed design) stage begins with the selected alternative conceptual design, refining and completing the design for that alternative. Much of the process is a more detailed application of the methods and processes used in the conceptual synthesis stage. Significant effort is spent in the selection, analysis and evaluation of all constructed elements of the structure.

- *Review.* The design review stage takes the completed detailed design and performs design evaluation, value engineering, simple redesign, and compliance checking of the functional and technical criteria; this involves spot checking as opposed to a complete checking of all governing criteria.

- *Construction.* The construction process converts the detailed design documents into a finished artifact. Currently the major emphasis of an integrated structural engineering CAD system is in the areas outlined above, but there are a number of useful functions which extend the application of the system into the construction stage, including the production of shop drawings, project scheduling, and construction management.[6]

Figure 2 illustrates the flow through the design process (the backward iteration flow paths in the process have been omitted for clarity).

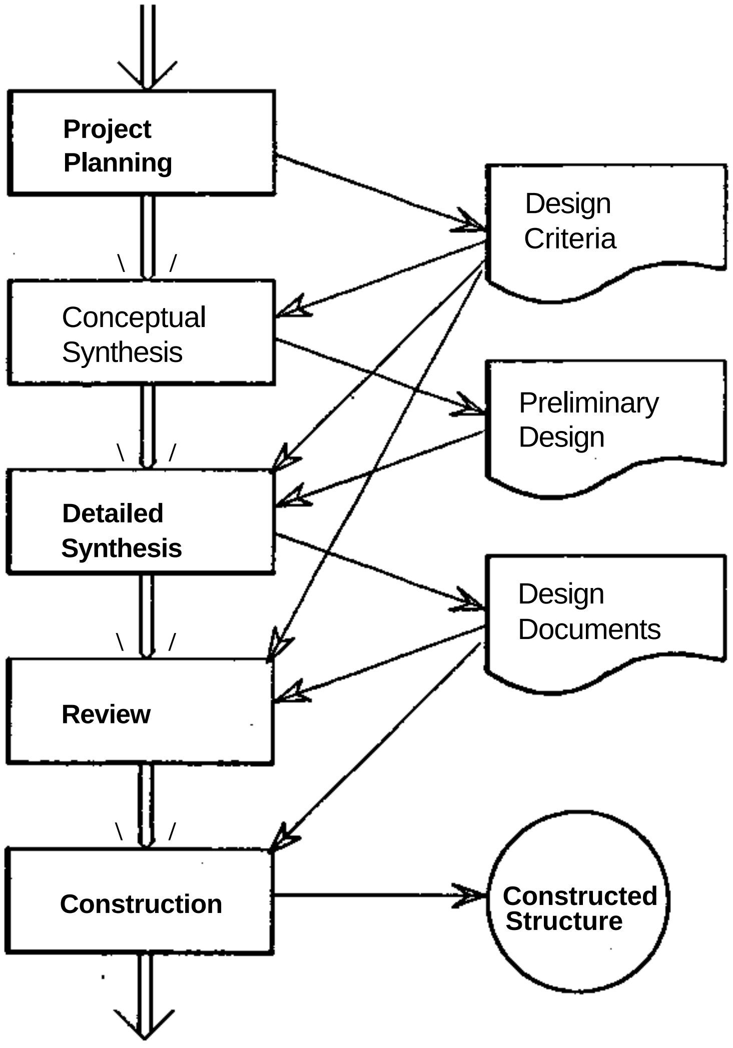


Figure 2: Schematic of the Structural Engineering Process

[6]Considerations for robotic based construction (i.e., *design for constructability)* appear increasingly important and will extend the integration of the design and construction stages, but are beyond the scope of this discussion.

Current computer use in the structural engineering process is extensive, but limited to algorithmic analysis, checking, scheduling, estimating and graphical presentation programs [Fenves 81]. Such tasks comprise only a small portion of the total design process — the tasks which have been programmed are usually intractable by manual means — with the remainder of the process being ill-structured and requiring knowledge based solutions. In addition, the process must deal with issues related to standards and the fragmentary nature of the construction industry.

2.2. Industry Fragmentation

As described above, the structural engineering process involves several distinct stages which are handled in a distributed fashion. There is a vertical separation among the stages; each stage builds upon the results of the prior stages, and the different stages often are performed by different individuals or firms. Similarly there is a horizontal separation among the various specialty professions and individuals involved in each stage. In the U.S., many of these firms have conflicting interests or desires and often interact in an adversarial relationship.

In terms of developing an integrated CAD system, the problem is not with the organizational structure of the participants, but with the handling of the design information. The structural engineering process involves the creation, manipulation and communication of the information which is the structural design. Due to the fragmented nature of the industry, the design information is also fragmented, and each discipline has its own *information model*. Information sharing is required to complete the design, and significant problems result from the informal mechanisms used in a manual communication process.

Centralized data handling and data communication mechanisms in a CAD system provide the capabilities for addressing the technical problems which result from construction industry fragmentation. However, a system must deal with the realities of the distributed process and the conflicting needs and goals of the participants.

2.3. Regulatory Process

Structural engineering is heavily influenced by the regulatory process. There is a multiplicity of standards, specifications, codes and regulatory agencies to be dealt with. Local code variants provide additional complexity. In addition to the mandated codes, the local shop practice and designer or user constraints are all of similar nature and structure, and it should be possible to deal with all such constraints with a common mechanism [Fenves 82a].

Standards[7] represent design knowledge and their use and application is an acquired task requiring expertise. Standards are written to be used in a passive compliance checking mode. However, provisions are often used as the basis of design rules, requiring reformatting and manipulation of the knowledge represented by the standard. There are problems related to application (which codes govern), selection (what are the applicable provisions), and interpretation and use (what the code means) of the standards [Rehak 81].

Algorithmic approaches to standards processing in current CAD applications have limited effectiveness [Stahl 83a, Stahl 83b]. Again, knowledge based and expert system approaches appear to provide the needed solutions for the successful use of standards in the structural engineering process.

[7]The term *standard* is used to refer to any type of design code, specification, or standard throughout the remainder. Standards processing is defined as the application, use and manipulation of standards in the design process.

3. Knowledge Based Applications in Structural Engineering

A number of knowledge based expert systems for structural engineering applications which have been built or are under development are discussed in this section. These systems are classified into three groups: those which perform design related tasks, those which deal with standards, and those which deal with the interface between design tasks and databases, paralleling the discussion of the issues in the preceding section.

3.1. Existing Systems

Several prototype expert systems have been built to perform various structural engineering tasks. These programs demonstrate the applicability of knowledge based processing techniques in the domain. The existing systems are all designed to address one specific step or task in the overall structural engineering process.

3.1.1. Design Tasks

Preliminary Design. *HI-RISE* [Maher 84] is an engineer's assistant for the preliminary structural design of high rise buildings (a more detailed description of *HI-RISE* is presented in a companion paper [Maher 84]). Starting with a specified spatial layout, the preliminary structural design process is divided into two ordered subtasks:

- Design of the lateral load resisting system
- Design of the gravity load resisting system

For each subtask, the space of design alternatives is represented by three dimensional subsystems, two dimensional subsystems, materials and components. The design process for each subtask consists of the following steps:

- *Synthesis* — Search for a set of feasible structural systems.
- *Analysis* — Perform an approximate analysis to insure the alternative under consideration provides the necessary load-carrying capacity.
- *Parameter Selection* — Select structural components by heuristics.
- *Evaluation* — Compute for each alternative a ranking based on a linear combination of factors such as drift, deflection, number of frames, member size, approximate costs and compatibility of subsystems.
- *Selection* — Select the *best* alternative based on a minimum value of the evaluation function.

HI-RISE is implemented as a rule based system in *PSRL* (Production Schema Representation Language) [Rychener 84a], a frame based production system built on *SRL* (Schema Representation Language) [Wright 83] (the development tools used in the C-MU environment will be discussed in more detail in section 5.8).

Analysis Consultants. *SACON* (Structural Analysis Consultant) [Bennett 78] is an early expert system for consultation and advice to non-expert engineers in the use of the *MARC* finite element analysis program. Based on a problem description, *SACON* recommends a modeling and analysis strategy. *SACON*'s knowledge base includes rules for inferring analysis strategies, rules for inferring controlling behaviors, and models for estimating behavior. Its inference mechanism is a backward chaining search of a goal tree. *SACON* is implemented in *EMYCIN* [vanMelle 81].

A similar system for the *SESAM-69* structural analysis package is *SESCON* [Fjellheim 83]. In addition, a second usage consultant for *MARC*, written in *FORTRAN*, has been developed [Rivlin 80].

3.1.2. Standards Processing

BRECON (Specification Consultant) [Sriram 81, Sriram 84] is a small prototype knowledge based system to aid in compliance checking of structural steel members to the AISC Steel Design Specification [AISC 80]. Previous work in computer representation and processing of design standards has used the decision table as the representation formalism for provisions [Goel 71, Fenves 73, Slirk 81]. Such decision tables can be readily recast into production rule or frame representations for knowledge based processing. The *SPECON* knowledge base is divided into two levels, rules which identify applicable constraints, and rules which represent the specific design constraints from the standard. Implementations exist in *LISP* and in *OPS5* [Forgy 81]. The inference mechanism is based on *MYCIN's* backward chaining strategy [Shortliffe 76].

The description of a *PROLOG* based knowledge based standards processor for the Australian SAA steel design standard has been presented [Gero 83].

3.1.3. Database Interface

HICOST [Howard 83] is used to develop preliminary cost estimates for high-rise buildings. It is used by *HI-RISE* to evaluate competing preliminary designs based on structure cost. Given the topology and geometry of a building and *HI-RISE's* preliminary choice for the structural system and elements of the superstructure, *HICOST* produces an estimate of the building's cost.

HICOST is implemented as a rule-based expert system, with algorithmic computational functions, which queries a database for specific data on component costs. It is a tightly coupled integration of several expert system tools and a relational database. The *cost estimator* is a hierarchically-organized production system with rules written in *PSRL.* Data objects are defined in frames in *SRL,* and cost data is stored in a relational database supported by *INGRES* [Stonebraker 76]. The production system computes the estimated cost from aggregate subsystem costs. Access to the database is through demons attached to *HI-RISE* cost schemas. Each subsystem data element has an associated cost value. Accessing the cost value of a basic component (beam, column, etc.) via a rule in the cost estimator triggers a demon which invokes a *FRANZ LISP* function [Foderaro 82] which in-turn calls a C procedure [Kernighan 78]. The C procedure uses *EQUEL,* the *INGRES* procedural query language, to access the database and return the item cost to the estimator.

3.2. Steps to an Integrated System — Research in Progress

In addition to the systems outlined above, the first knowledge based systems designed to address more than a single engineering problem solving task are under development. Such systems form the basis for an integrated knowledge based structural engineering CAD environment. Since many of the concepts used in these systems provide the basis for the architecture of the integrated system presented in Section 5, only a short introduction to the components is given **below.**

3.2.1. Design Tasks

DESTINY (Integrated Structural Design) [Sriram 83] is an extension of *HI-RISE* to encompass more building configurations, more design tasks and to provide a more flexible design process. The system consists of a number of *knowledge modules* (KMs) (which are comprised of smaller rule sets) which communicate through a blackboard. Its structure is modeled after expert system frameworks such as *HEARSAY-II* [Erman 80] and *HASP/SIAP* [Nii 82]. In *DESTINY,* knowledge is divided into four levels:

- *Strategy KMs* analyze the current design state and determine the next action.
- *Activation KMs* invoke the appropriate specialist KMs to carry out the current design strategy.
- *Specialist KMs* perform one knowledge based processing task, and are similar to traditional expert systems. Specialist include:

- o *Preliminary Design System* used to generate potential structural configurations *(ALL-RISE* — an extension of *HI-RISE).*
- o *Analysis and Modeling Consultant* used as a modeling and result interpretation interface for a finite element analysis program (similar to *SACON).*
- *o Specification Consultant* used to interface to **a** set of design standards (similar to *SPECON).*
- *o Estimating Consultant* used to determine cost estimates for alternative designs (similar to *HICOST).*
- *o Design Critic* **used to evaluate alternative designs based on functional and technical criteria.**

• *Resource KMs* provide the knowledge based and algorithmic processors required for design and analysis such as: finite element analysis, database management systems with associated design and component databases, and standards processors and associated standards.

The overall structure of *DESTINY* is depicted in Figure 3. *DESTINY* is **currently under development and is being built from several tools including** *PSRL, INGRES,* and *FINITE* **[Lopez** *77],* **and** is **being written in several programming languages including** *FRANZ LISP, C* **and** *FORTRAN.*

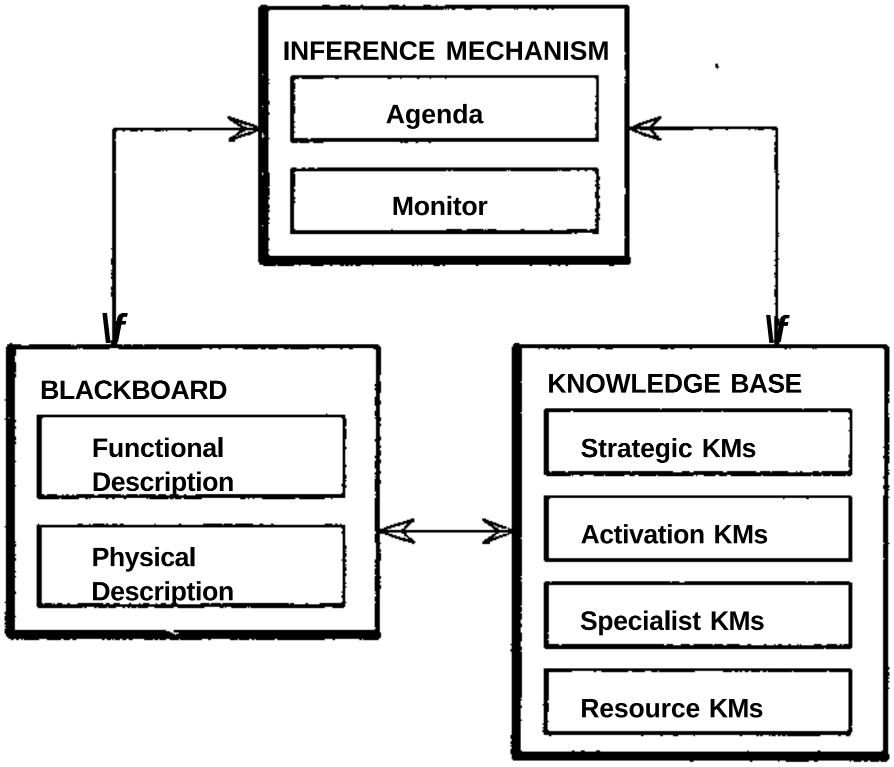


Figure 3: ***DESTINY*** **System Structure**

3.2.2. Standards Processing

SICAD **(Standards Interfaces in CAD) [Lopez 84a, Lopez 84b] is a knowledge based (non-expert) approach to standards processing. Rather than recasting the common decision table formulation of standards into rules,** ***SICAD*** **uses a custom knowledge base and inference mechanism which accesses and uses standards represented in their common decision table formulation. The knowledge base contains three items:**

- *Classifier Trees* are used to relate engineering terminology to provisions of a standard.
- *Information Network* is a network of decision tables which represent the provisions of an standard.
- *Mappings* are used to relate data items in a standard to data items in a design database.

Of these, the decision table provisions are represented as algorithmic code, and the remainder are stored in the knowledge base. The system is written in *FORTRAN* and uses *POLO* [Lopez 72] for database and knowledge base support.

SICAD is designed to be interfaced with a design program which needs to perform compliance checking. The design program invokes *SIQAD* to identify and check applicable provisions. Checking uses a goal driven search strategy to determine all data items required to evaluate a provision. Checking will automatically access this data in the program's database through the mappings stored in the knowledge base. The structure of *SICAD* is shown in Figure 4.

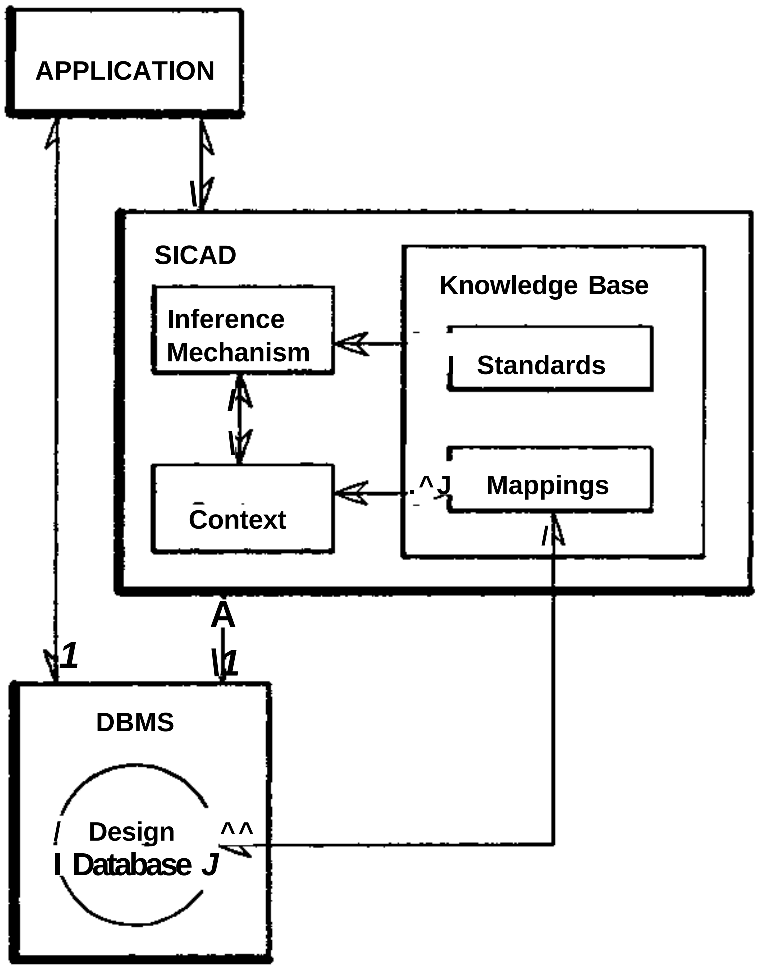


Figure 4: *SICAD* System Structure

3.2.3. Database Interface

KADBASE (Knowledge Aided Database Management System) [Howard 84] is a multiple database manager for knowledge based structural engineering applications. It provides the capabilities for an integrated system to access a collection of databases, and it addresses the issue of interfacing databases with expert systems. Its overall architecture resembles a networked heterogeneous database management system [Cardenas 80]. The individual databases may be based on different data models (relational, network, hierarchical) and may reside on a different host computers. However *KADBASE* is not concerned with the issues involved in the physical networking of the databases.

In the current applications which combine databases and expert systems, the expert system is tightly coupled to the database in one of two ways, either the expert system has detailed knowledge of the syntax and semantics of the database [Blum 82, Howard 83], or the expert system and database form a combined system [Kellogg 83]. *KADBASE* is an attempt to provide a flexible interface between expert systems and networks of databases by freeing the expert systems from the requirement that they contain detailed syntactic and semantic database knowledge.

KADBASE provides an interface which allows an expert system to issue queries and updates and receive replies, all in terms of its own information model. In addition, each individual database communicates with the interface in the context of its own database schema. The interface uses mapping information provided by the individual expert systems and database schemas to construct a *global* schema. Queries from an expert system are first translated into the context of the global schema. *KADBASE* determines a strategy for answering the query within the context of the global data model and decides which databases contain the necessary information to answer all or part of the query. Subqueries to the individual databases are translated from the global data model to the syntax and semantics of the design databases. The replies are handled in the reverse fashion. In each translation, *KADBASE* performs a two level, two step language transformation process to communicate between the expert and database management systems (the translation process used in *KADBASE* is discussed in more detail in Section 5.5). The basic system architecture is shown in Figure 5.

4. Issues in Building an Integrated Knowledge Based Structural Engineering System

The research projects described above represents some of the first steps towards building an integrated system for structural engineering applications. The following discussion outlines some of the specific issues which such an integrated system must address and provides some thoughts on how such a system will evolve.

The basic architecture of the system is based on a highly distributed and loosely coupled set of programs. Given sufficient resources, the design would probably be different. Building a complete integrated system with one overall designer, one consistent design philosophy, and one data and knowledge representation strategy should yield a better system, due to consistency. Sufficient resources to construct such a system are not available. Thus the system design philosophy is to accept and integrate a variety of different components from various sources. Not only does this provide a mechanism to build on the work of others, it lets the system evolve over time and meshes with the reality of the distributed nature of the structural engineering process and the emerging trend towards distributed processing.

As discussed above, the basic issues which the system must address relate to the structural engineer-

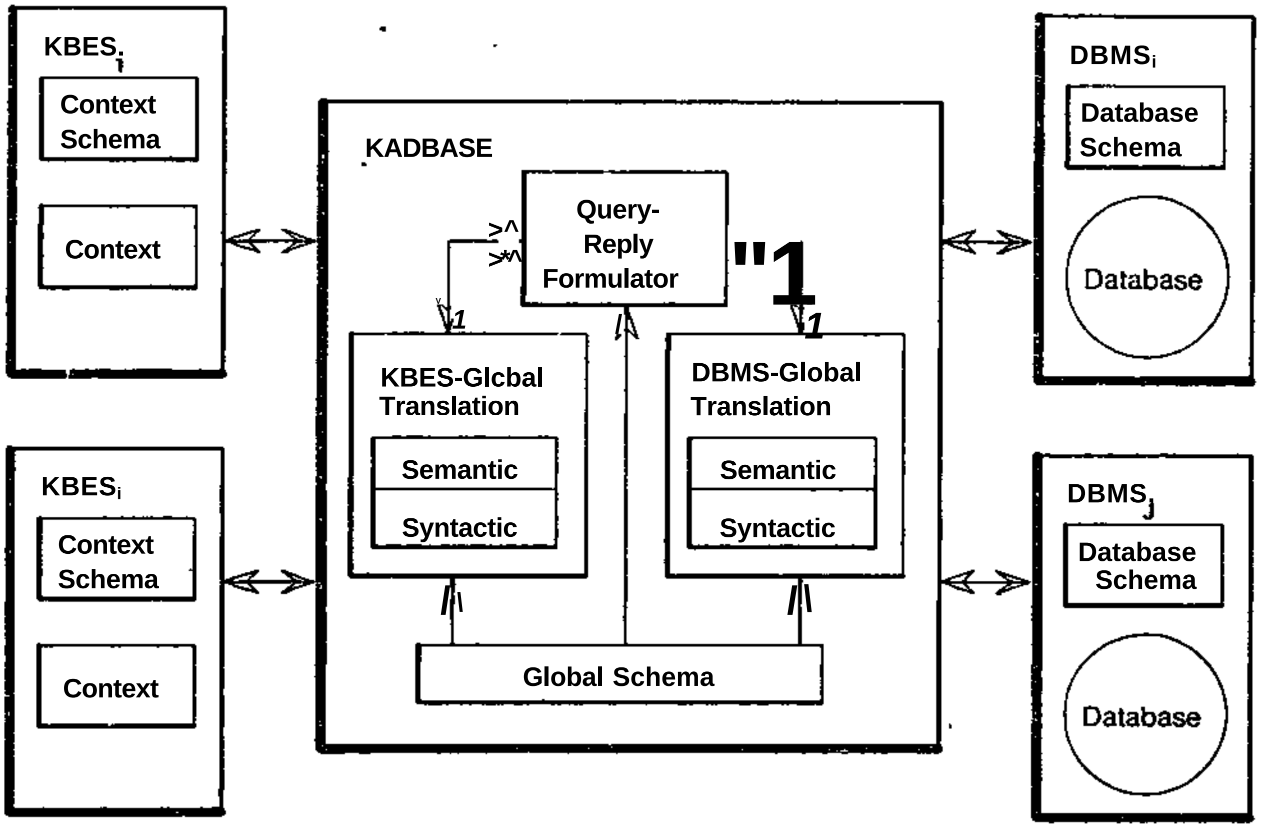


Figure 5: *KADBASE* System Structure

ing process, the use of standards in design, and providing database and information communication support to coordinate the various individual participants in the design process. The remainder of this section outlines a set of capabilities which an integrated structural engineering CAD system must possess and problems that must be addressed.

4.1. Integrated Knowledge Based Design Task Processors

There are a variety of tasks which are performed during the structural design process, as outlined in Section 2 and 3. For each of the tasks, a knowledge based expert system could be built to aid in the task, acting as either an advisor, consultant, critic, or design assistant.[8] These tasks form the basis of the system. In the domain of high-rise building design, some of the task-specific knowledge based processors are listed in Table 1. The list is only representative. Any number cf synthesis, design, analysis, and evaluation tools for all aspects of the process could be built. A similar list of knowledge based tools could be formulated for other structural forms such as bridges, off-shore platforms, power plants, etc.

Consideration of the types of tasks represented by the list in Table 1 yields a set of issues related to the design of the components of the system, as described below.

- *Unknown Tool Set.* Only a few prototypes of the knowledge based tools described above exist. Since knowledge based technology is invaluable for developing computer aids for ill-structured

[8]All the tools act as computer based aids. They are not designed to replace the engineer, only to augment and assist

Planning Consultant	*Architectural Design System*
Spatial Layout Consultant	*Site Planning Consultant*
Preliminary Structural Design System	*Approximate Structural Analysis System*
Finite Element Modeling Consultant	*Finite Element Analysis System*
Component Design System	*Geometric Modeling Consultant*
Substructure Design System	*Substructure Analysis Consultant*
Electrical Distribution Design System	*Electrical Distribution Analysis Consultant*
Mechanical Design System	*Mechanical Design Analysis Consultant*
HVAC Design System	*HVAC Analysis Consultant*
Vertical Transportation Design System	*Cost Estimating Consultant*
Regulatory Compliance Consultant	*Design Critic*

Table 1: Knowledge Based Tasks in High-Rise Building Design

design tasks, many additional tools can be anticipated. However, because a complete integrated structural engineering environment does not yet exist, the complete list of system components cannot be fully anticipated in advance.

- *Diverse Tool Set*. One group will not be responsible for developing all of the tools. Thus it can be expected that different tools will have different operational philosophies and will be built using different expert system frameworks. It is likely that the prototype versions of many tools will be designed to operate in a stand-alone mode; because of the scope, and following the fragmentary nature of the industry, each of the programs should be considered as an autonomous system. As expert system building frameworks evolve, the complexity of the resulting programs in expected to increase. Systems like *DESTINY*, which are built from cooperating, intelligent processes, should become more common.

- *Role of Algorithmic Applications*. Certain tasks, such as approximate structural analysis, or finite element analysis, are best performed by an algorithmic processor. Excellent algorithmic programs exist for such domains, and their redevelopment as knowledge based systems does not appear useful or beneficial. Thus, support mechanisms to use and access a variety of algorithmic programs are needed.

- *Knowledge Sharing*. Certain problem solving tasks are common to one or more design tasks. For example, electrical, mechanical and HVAC systems must all reason about the spatial and structural configuration of a building. Similarly, both structural and substructure design tasks need to reason about structural loads. More complex expert system tools, such as those using blackboard models, use knowledge sources to represent parts of the problem solving process. These are similar to procedures in an algorithmic program, and just as repeated use of common procedures in multiple programs can reduce development effort, repeated use or sharing of knowledge sources is valuable.

- *Integration*. Engineering is a cooperative effort, with the design being built over time. A number of individual knowledge based processors will be required to complete the design. Thus, there must be a mechanism for the various processors to cooperate; the mechanisms to integrate the individual pieces into a single design environment must exist. With such integration mechanisms in place, the development and use of autonomous system components (which are integrated into the full system) is natural. Complete integration of the individual tools into a complete system could be performed manually. However, in building an integrated system, one can anticipate the same difficulties in dealing with data representation and inter-program communication as those which arise when algorithmic tools are integrated and linked to database management systems.

- *Standards and Database Interfaces.* Use of standards and design and component databases are an important part of the structural engineering process. To maintain flexibility, an integrated system must include appropriate interface mechanisms for accessing and using these components. Just as the set and nature of the tools cannot be determined a priori, the set of standards and databases which comprise the system cannot be determined. A flexible, loosely coupled interface between the components in required. Issues related to this coupling will be discussed in Sections 4.2 and 4.3.

- *Strategic and Tactical Processing.* The structural engineering process can be envisioned as a multi-level process. At the highest level is a knowledge based strategic supervisor which decides *what* actions should be performed *when.* It invokes lower level tactical components which actually perform design tasks. This multi-level organization is identical to the multi-level hierarchy of senior designers and staff engineers in engineering design offices. Again, at the tactical level, there is a two part hierarchy of determining what to do, and taking actions to do it. The tools listed above are some of the specific tactical components of an integrated system, and they do not address the global problem solving process. A strategic controller is needed to integrate the other components into a complete system and to perform the supervisory role.

- *Design Tool Declarative Knowledge.* Just as schema information makes databases self descriptive, other components could be made self descriptive. Knowledge bases and contexts could include schema information on logical content and organization. Thus, rather than a programmatic linking of components, intelligent interfaces which use knowledge about the components being linked to perform communications and data sharing are possible. Similarly, metaknowledge which describes the problem solving process and applicability of any design program could be added to its knowledge base. Again, rather than programmatically integrating all components through a hand crafted, tightly coupled design process controller, an knowledge based processor which uses the metaknowledge about the design tools capabilities and features could be used to dynamically select the global problem solving strategy. A similar approach appear useful for interfacing to databases and standards processors.

4.2. Standards Processing

SICAD and *SPECON* both treat standards as knowledge, and demonstrate that knowledge based standards processing is feasible. They address a number of key issues, but a number of problems related to the use of standards in an integrated system still remain. These issues are described below.

- *Interpretation.* Problems associated with interpretation of individual provisions and relationships between provisions are eliminated when standards processing software is used. Tools such as *SASE* (Standards Analysis, Synthesis, and Expression) [Fenves 79a, Fenves 79b] **for** formulation and organization of standards exist; providing a single common formalism **for** representing standards. Use of these tools by standards[1] writers yields a machine processable form of the information network, classifier tree, and decision tables representing the standard, that does not require reinterpretation for inclusion in a program, as does the use of the common text form of the standards. This declarative knowledge can be directly translated into the format of the standards processor's knowledge base without semantic interpretation. However, such a translator does not yet exist, and most standards do not exist in machine processable form.

- *Changing Standards.* Changing a standard, which invalidates an algorithmic program, is conceptually as simple as changing a knowledge base used by the standards processor. A program can access the changed standard without modification; however, it may not function correctly if it contains an implicit semantic interpretation of the standard.

- *Data Access*. *SICAD* provides a mechanism for the standards processor to automatically access design data from an application program's database. Mapping functions relate information in the standards processor's context to corresponding database information. Thus the application is not explicitly linked to the standard. Also, the description of the standard remains *generic* and does not contain any database references. In *SICAD*, the database mapping functions must be hand coded and reside in the standards processor's knowledge base. Changing either a standard or a database may require some rewriting of the mappings.

- *Application Access*. *SICAD* and *SPECON* assume an application program (either knowledge based or algorithmic) invokes the standards processor. Both systems provide a generic mechanism to access any standard without a tight coupling of the data spaces of the application and the standards processor. However, the application must still contain explicit references to the standards processor. A system like *DESTINY*, which relies on the two processors being at different levels, further uncouples the two components.

- *Active Constraint Processing*. The current knowledge based standards processors perform only *passive* processing. The search strategy of the inference machine provides the mechanism to determine the applicable provisions in a given context and to check a component against those provisions. Designer often use provisions as the basis for design rules, selecting components so that they are acceptable but not over-designed. To support such *active* processing, three additional capabilities are needed in the standards processor:

    - Design is not an exhaustive use of all applicable provisions. Some application independent method of determining governing provisions is needed. Additional heuristic knowledge on the content, use and structure of a standard could be used to augment the declarative knowledge representation currently used.

    - Design rules result from converting the checking form of a provision into an assignment form for a particular variable. *CONMAN* (Constraint Manipulator) [Holtz 82] is an example of a symbolic processor for such conversions.

    - When a provision check fails, it is necessary to know *why*, i.e., to know what to do and how to change the design so that the provision will be satisfied. Again, heuristic knowledge about the use of the standard is required.

- *Unified Approach to Constraint Processing*. Other types of design constraints, such as designer *style* or local shop practice, are similar to standards in their structure and use. All forms of constraints can be treated as knowledge, with a single inference mechanism used for all types of constraint processing. This area has not yet been explored.

4.3. Database Interface

An integrated *data space* forms the basis for integrating the various processing components into a single system. This data space is represented as a collection of engineering design and library databases. Engineering design databases are different from commercial databases, in both their structure and use [Eastman 81], and a detailed discussion of database needs and capabilities is outside the scope of this paper. In an integrated knowledge based processing environment, the following key database issues related to interfacing and data access must be considered.

- *Distributed Databases.* As discussed above, the complete integrated system will be built from a variety of individual programs (both algorithmic and knowledge based). Each of these will have their own data structures, databases and information models. Developing a central common database will be unlikely; thus the integrated system will need to deal with distributed databases. Using distributed databases raises new problems of consistency maintenance, data placement and data backup.

- *Integration and Interfaces,* In the desire to maintain functional and logical independence of the design tools from the databases, some form of database-to-design tool interface (similar to *KADBASE)* is needed. This permits the design tools to remain generic; all of the detailed database access information is associated with the interface instead of being associated with the individual design tools. The problem is similar to interfacing the standards processor to the design databases as described above.

- *Database Declarative Knowledge.* In order to provide the required flexibility in the interface, it is necessary to reason about the content of the databases and the data space of the design tools which are being interfaced; the interface must determine where data resides and how it is stored. The information used in this process can be expressed as declarative knowledge about the data schemas: their syntax, semantics and use. A knowledge based processor can then provide the interface functions.

- >AcfVe *Database Processing.* A database need not be a passive information repository [Fenves 82b]. Associated with data items are constraints on their values. These constraints provide more than integrity enforcement; they can be used to associate active design processing procedures with data items [Fenves 82a]. Proper mechanisms for dealing with the design constraints in the database must be developed.

5. Conceptual Architecture of an Integrated Knowledge Based Structural Engineering Environment

This section discusses a conceptual view of a comprehensive integrated CAD environment which incorporates knowledge based processing throughout the entire structural engineering process. The system design attempts to address the issues discussed above.

In the previous work [Rehak 81], the overall architecture of an integrated civil engineering CAD environment was presented at a conceptual level. No details on hardware or software configuration, or on knowledge representation or expert system framework were presented. The following discussion updates the *CAESE* architecture and illustrates current thinking on the actual implementation and design of the components of such a system.

The overall system architecture is a collection of loosely coupled processing components operating on a distributed heterogeneous hardware base. The components consists of active processors used to perform design and engineering, such as inference machines, databases, etc., and development tools, such as knowledge acquisition modules. The basic components of the system are similar to those proposed for *CAESE.*

- A design strategy controller to oversee the entire structural engineering process and respond to the user's requests for design assistance,
- A set of knowledge based design processors *(design tools)* (such as *ALL-RISE, SACON, HiCOST,* etc.),

- A knowledge based standards processor and associated knowledge bases containing applicable standards,
- An engineering database management system and a variety of databases containing catalogues and design information,
- Algorithmic processors (such as finite element analysis), and
- A sophisticated user interface system based on personal engineering workstations.

The user interacts with the system through a workstation, requesting some form of processing or design assistance. The design controller responds to the user's request and selects a strategy to provided the requested information. Some subset of the design tools will be invoked to perform the needed computations, and these will interact with other design tools, the standard processor and the database management system to complete the task.

The following section presents the generic design of a blackboard model knowledge based framework used for components of the integrated design environment. The next three sections describe the specific knowledge based components (design controller, design tools, standards processor and database interface) of the system which are built using this basic blackboard model framework. This is followed by a presentation of the distributed communications mechanisms and the user interface. The discussion concludes with a presentation of the hardware and software currently being used for prototyping the system. If the framework described in the subsequent sections was available, it could be used to implement knowledge based CAD systems in the structural engineering domain.

5.1. Blackboard Model Knowledge Based System Framework

In addition to the knowledge based processors used for the various design tasks (the design tools), several *system level* knowledge based components are required, including:

- the overall strategic design controller;
- the database-to-design tool interface; and
- the standards processor.

The current choice for an expert system development framework for both the design tools and the system level tools is a blackboard model framework [Balzer 80, Erman 80, Nii 82]. This framework provides an excellent paradigm for knowledge intensive problem solving that requires multiple cooperating, communicating intelligent problem solvers. As such, it parallels the exact nature of the structural engineering problem solving process.

The problem solving behavior is based on coordination of individual processors denoted *knowledge modules.* Each of these is developing a different hypothetical solution to the problem (or some subpart of the problem) at a different level of problem abstraction. Communication is required to generate, combine and evaluate the various hypothetical solutions. The central data structure (the context) used to support this communication is denoted the *blackboard.*

The basic architecture of the blackboard model knowledge based system framework is depicted in Figure 6. For this discussion, the pertinent aspects of the framework are:

- *Knowledge Base.* Knowledge is aggregated into sets of independent knowledge modules (KMs) each of which address one subproblem. These KMs can be divided into two basic classes:

    o *Strategic KMs* are used to control the problem solving process. The strategic KMs

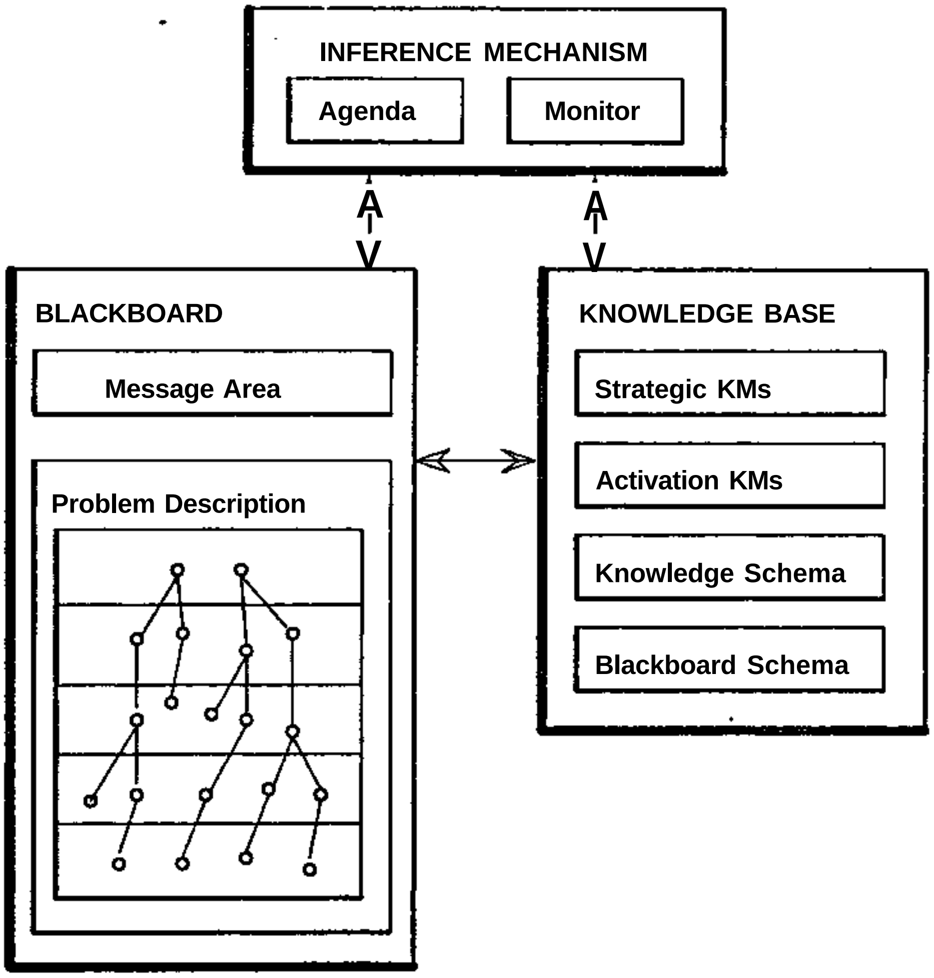


Figure 6: Blackboard Model Knowledge Based System Framework

analyze the current state of the solution process and determine what the next course of problem solving action should be and the processing necessary to initiate these actions.

o *Tactical KMs* are used to carry out the actions planned by the strategic KMs. The tactical KMs perform one knowledge based problem solving task or subtask. Each may use a combination of heuristic and causal knowledge and may interface to algorithmic processors or other design tools.

Each of the KMs is essentially a mini expert system dealing with one aspect of the entire problem. The overall organization of the blackboard system does not set any requirements on the development framework used for the KMs; they may be rule-based, frame-based, network-based, etc., or they may be simple interfaces to algorithmic programs.

In addition to the KMs, the knowledge base includes *knowledge sources* (KSs). Unlike KMs, KSs do not perform active processing; they contain declarative knowledge and data used by the KMs, such as the meta-knowledge describing the KMs.

A special class of KSs form the *knowledge schema,* analogous to a database schema. They provide the mechanism to store the meta-knowledge describing the KMs. This metaknowledge includes how KMs are used, when they are activated, and how they relate to the context (data required and results produced) and to other KMs. The meta-knowledge is declarative knowledge describing the problem solving process, and it is used by higher level strategic processors which determine what processors should be applied at what time.

- *Blackboard.* The context is implemented as a blackboard which contains a message posting area, information about the state of the problem solving process and a description of the current design. The message area provides a mechanism for the various KMs to communicate and place explicit requests for information and actions. The blackboard has a static organization containing data slots for all KMs, and is organized into levels which represent different levels of problem description abstraction. These slots contain descriptions of the design hypothesis and design alternatives currently under consideration. As the solution proceeds, the various KMs independently generate, update and evaluate the solution hypotheses.

  In addition, associated with the blackboard is a specialized knowledge source, the *blackboard schema,* which contains the declarative knowledge describing the content, organization and use of the blackboard. In addition to describing the structure of the blackboard, it contains a description of which KMs deal with the various slots. This information is used to interface (and integrate) the various components in the system. It provides other processors, such as the database interface, with a mechanism to interpret the blackboard and to post new data on the blackboard.

- *Inference Mechanism.* The basic operational strategy of the system is to propose and attempt to prove design hypotheses. The various KMs determine how to formulate and prove the hypotheses (strategic level) and how to proceed with steps towards hypothesis verification (tactical level). The inference mechanism consists of two components: the *agenda* and the *monitor.*

  The agenda is the list of processing tasks which the system is currently considering. At any time the agenda contains all tasks to be performed based on the data on the blackboard. To permit communications between the KMs, the agenda is stored on the blackboard.

  The monitor is used to track problem solving, place tasks in the agenda and to heuristically select the next task for execution. It checks the blackboard and the message area after the execution of each KM, examining the slots on the blackboard associated with the hypotheses which define the problem solution state. Based on either changes to the data (event driven) or an attempt to verify a hypothesis (goal driven), the monitor determines which KMs are applicable in the current state. These are combined with requests from the message area, and are rated and combined with the current agenda. From this list, the monitor selects an action (KM) to be invoked. Based on the problem solving progress, as monitored by strategic KMs which are scheduled after changes to the blackboard, the priority of agenda items are changed, providing the means to control the problem solving process. The monitor's processing strategy (heuristic selection, task monitoring, focus of attention, etc.) is not predefined, but is specified through a set of rules.

5.2. Design Controller

The design controller is the highest level knowledge based processor in the system. Its strategic KMs contain the heuristic knowledge of how to conduct the structural engineering process, i.e., the role of and relationships between the various steps and data items in the design process. The design controller's blackboard contains a description of the current state of the *global* problem solution.

An engineer initiates, via a workstation, some aspect of problem solving by directing the design controller to perform some task or to start a specific design tool; the design controller does not perform any of the steps actually used to design the structure. The strategic KMs monitor the overall process and decide what actions should be performed to complete the engineer's request, determining which tactical KMs are available for solving the problem and posting blackboard requests to invoke these KMs. Other strategic KMs then evaluate the proposed alternatives and proceed to invoke and monitor a set of tactical KMs to perform the actual problem solving.

Each of the design controller's tactical KMs is composed of two parts:

- *Design Tools* are the problem solving components of the integrated system. Each is a separate knowledge based or algorithmic processor. They are not an integral part of the design controller, but each is an individual program operating on some processor of the distributed hardware base.

- *Design Tool Controllers* provide the interface between the design controller and the individual design tools (one tool controller per design tool). Each provides the mechanism (message passing) for the design tool to communicate and share information with the controller, and it monitors the execution of the design tool.

Adding a new design tool to the system requires a tactical KM *design tool controller* be written and added to the knowledge base of the design controller, and the strategic KMs and knowledge schema be augmented to include knowledge about the capabilities and use of the design tool. Conceptually, the system architecture is designed so that no other interfacing is required.

5.3. Design Tools

Each design tool (algorithmic or knowledge based) performs one problem solving task or subtask. As outlined above, each is a standalone process operating under the supervision of the design controller.

Since many design tools have a complex structure, it is anticipated that newly developed knowledge based design processing tools will be based on the blackboard framework, and will be designed to fit into the overall system structure.

A processor such as *DESTINY* fits directly into the model framework described above (*DESTINY*'s Strategy and Activation KMs are at the strategic level, and the Specialist and Resource KMs are at the tactical level). Tactical processes, such as finite element analysis, standards processing and design databases, are coupled into the tactical level of *DESTINY*. In the integrated environment, these processes are not tactical KMs of any one knowledge based design tool; each is a separate design tool.

For an in-house program like *DESTINY*, it should be possible to modify the tool to incorporate the necessary knowledge base and blackboard schemas required for integration. For imported applications this may not be feasible or desirable. A small domain independent tool kernel can be built which contains the context and knowledge base schemas and a simple strategic KM. This strategic KM communicates with the corresponding design tool tactical KM in the design controller and invokes one tactical KM in the design tool. The imported application itself becomes the design tool tactical KM. For each imported application, the domain independent tool kernel can be instantiated with the necessary context and knowledge based schema information to yield a complete design tool. Thus, at the design controller level, all tools appear identical. The structure of such a kernel design tool is shown in Figure 7.

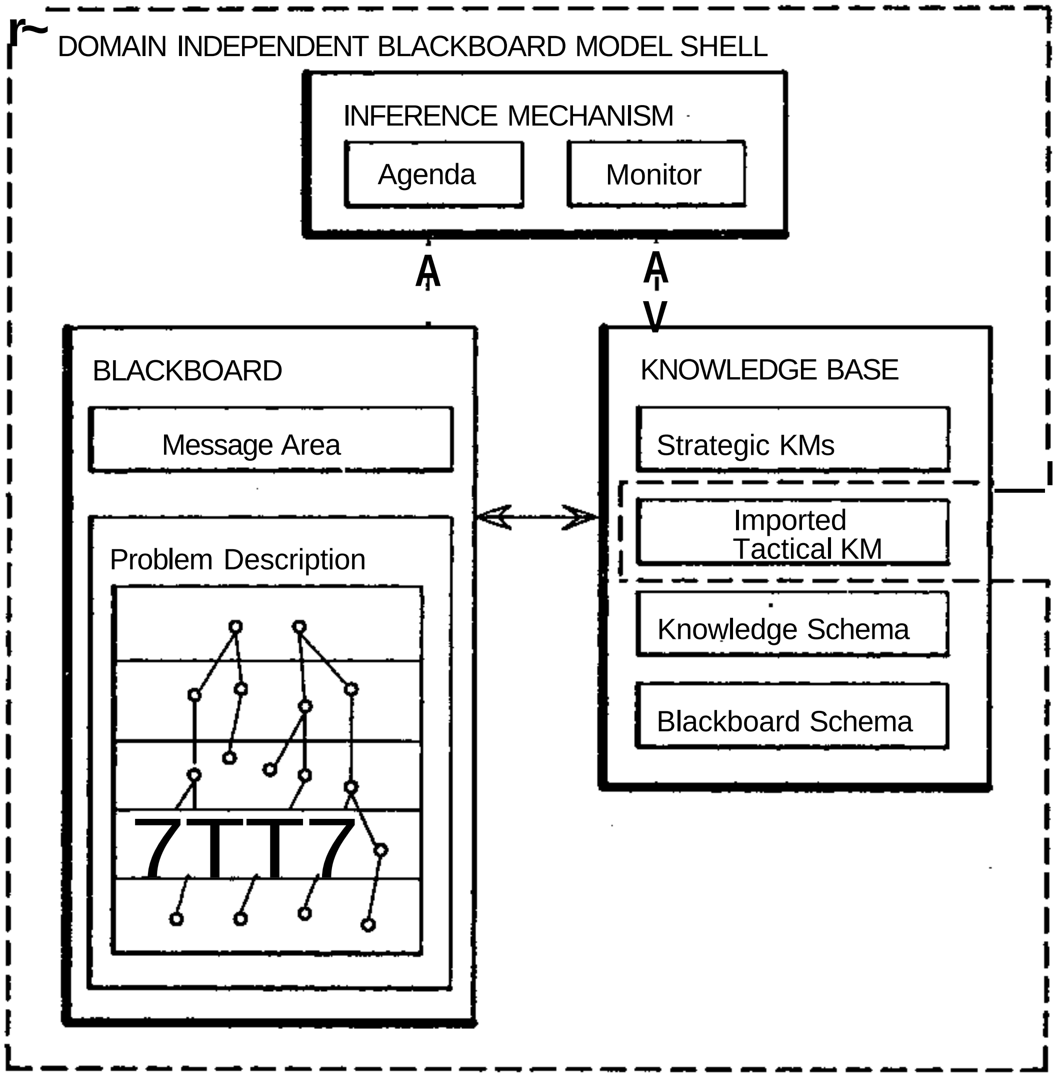


Figure 7: Kernel Design Tool Structure

5.4. Standards Processor

The standards processor is a knowledge based system which contains subprocessors with capabilities for access to and checking of provisions as in *SICAD,* for manipulation of provisions as in *CONMAN,* **for** active processing and for heuristic access. The processing capabilities provided by *SICAD* and *CONMAN* are independent of the actual standards to which they are applied and can be implemented as components of a single processor. The capabilities are provided through a set of KMs:

- KMs which implement the goal driven search strategy for applicable provisions as in *SICAD.*
- KMs which provide the mechanism to evaluate any provision.
- KMs which provide active processing to recast checking provisions into design rules as in *CONMAN.*

For each individual standard, the knowledge base contains data which is dependent on the individual standards:

- KSs which are the declarative descriptions of the standard (i.e., the classifiers, information network and decision tables).
- KMs which implement a heuristic search strategy.
- KMs which implement active processing capabilities.

One copy of the standards processor will be instantiated for each specific standard used in the system. Each standards processor combines the standard independent KMs with the standard dependent KMs, forming one unique processor. Selection of the actual standards incorporated in the system will be dependent on the structural engineering application domain of the system (bridges, buildings, etc.). In addition to the active standards processor components, the standards processor knowledge acquisition module will include a *SASE* like system for manipulating and checking the organization, correctness and structure of the standards. A schematic view of the standards processor is shown in Figure 8.

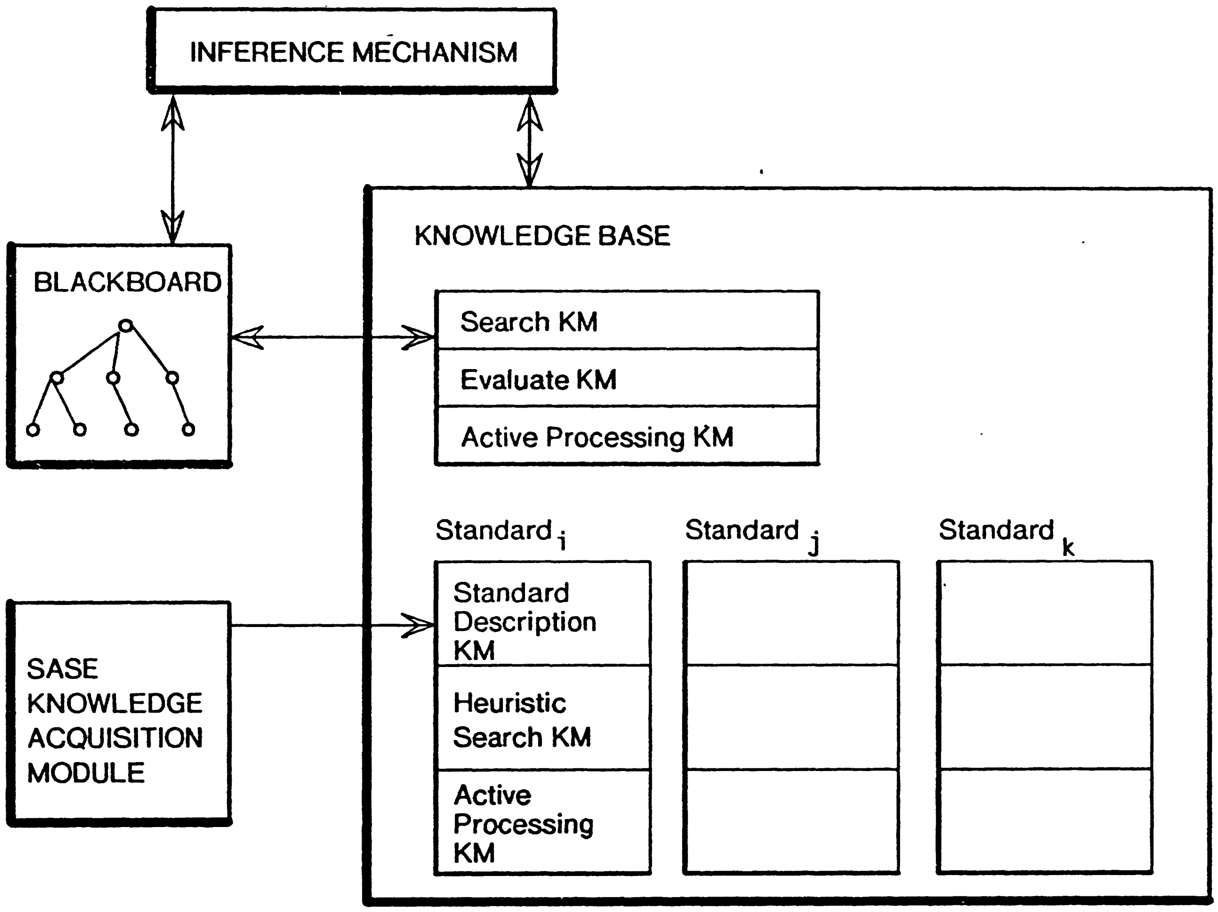


Figure 8: Standards Processor Structure

5.5. Distributed Data Interface

KADBASE provides the format of a knowledge based distributed design database management system. Such a loosely coupled interface requires knowledge describing the data spaces of the components being interfaced. Each basic database management system provides data access mechanisms and database schema information, and supports a number of databases. Additional knowledge describing the syntax and semantics of the database interface is needed for each database management system. For each database, semantic knowledge of the database schema also is needed. This information is provided by tactical KMs which perform syntactic and semantic translations between a database and the *global* data space. The blackboard schema information associated with each knowledge based processor provides similar information. Again, tactical KMs provide the translation between the global data space and the local data space of the knowledge based processor. Although used by the .nterface, the knowledge describing each component resides with that component. Thus the combination of the database knowledge arid a basic database management system forms a knowledge based database management system. Such a component is similar to all other knowledge based design tools; the basic database management system is a tactical KM.

Database access for a design tool (or any other component of the system) is provided by a knowledge based interface processor using the blackboard and database schema knowledge. The interface processor's knowledge base contains a *global* data space schema. A design tool's request for data is passed to the interface (details of the communications between the components are described in section 5.6). The interface uses the tactical translation KMs associated with the design tool to first perform a syntactic translation of the request into the global schema, followed by a semantic translation. A strategic KM of the interface then uses semantic knowledge of the individual databases to determine where to locate the data *[target* databases), and to form a set of queries. Using the translation KMs of the databases, each subquery is semantically and syntactically translated for the target database. The new requests are then passed to the target database management systems. Replies to the requesting design tool are handled in the reverse fashion. During this reply process, a strategic KM aggregates subquery responses and formulates the final reply for the design tool. The structure and relationship of these components is shown in Figure 9.

The process is not limited to interfacing between a design tool and a database. A similar interface process can be used to pass data between any two processes based on their data or blackboard schemas and translation knowledge.[9]

5.6. Communications Server

As outlined above, each of the individual components operates as an autonomous process under the direction of the design controller. Communications and data sharing is required to integrate the tools and to provide for their cooperation in problem solving. Each system component has a communication port, through which messages pass to and from the design controller. Tasks can be linked in two ways:

- *Explicit Linkages* in which a KM that requires a certain service knows of another design tool which provides the service. A message is passed to the design controller requesting the explicit service of the design tool. The controller processes the request, sending the request to the target tool which performs the function. Once the request has been completed, the design controller is informed, and it sends a reply to the KM which requested the service.

---

[9] H~he schema and translation (semantic and syntactic) knowledge for any component forms its *information model.*

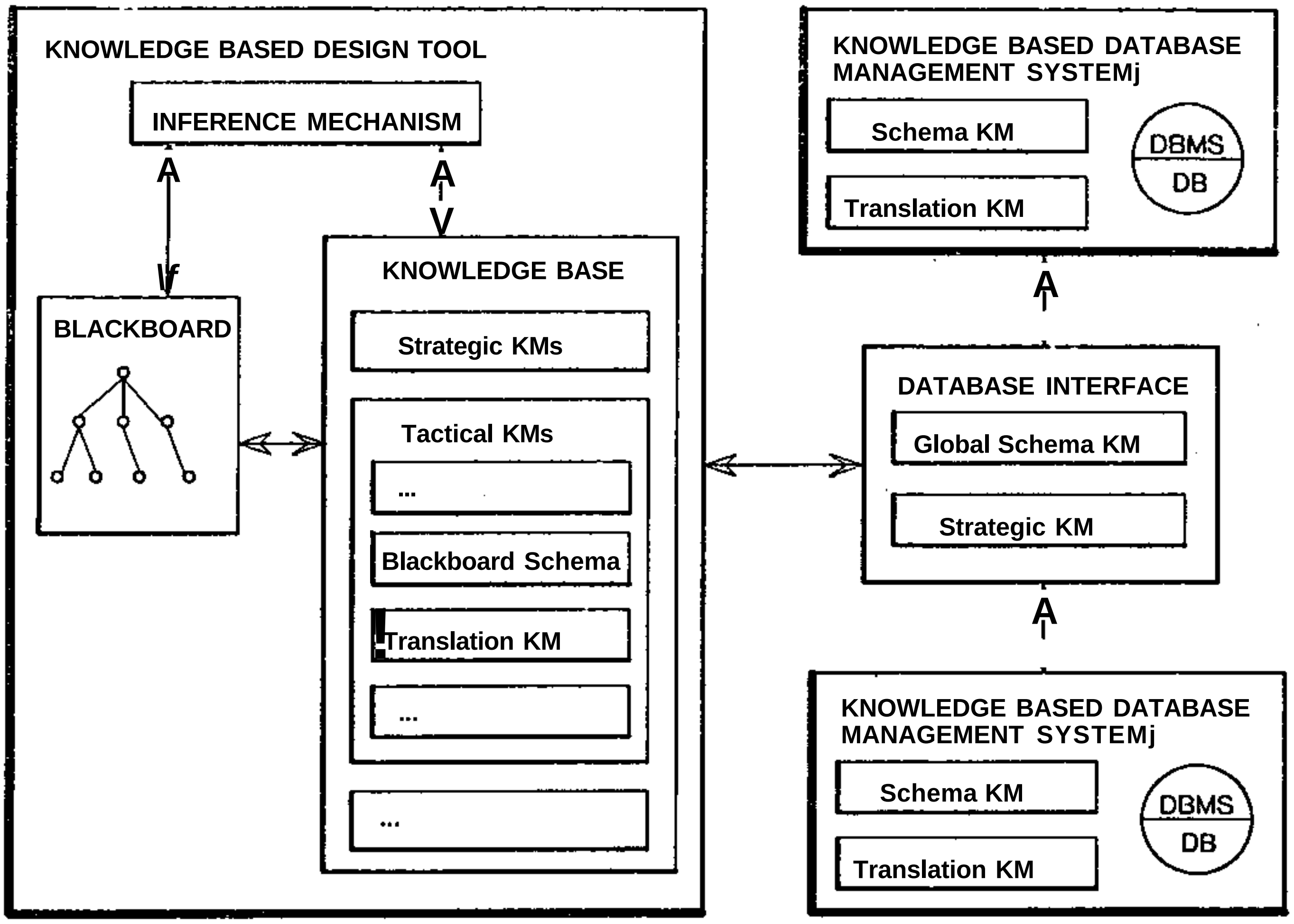


Figure 9: Database Interface Structure

- ***Knowledge Based Linkages*** **in which a KM requires a certain service but does not know who provides the service. Again, a message is passed to the design controller requesting the service. Using the meta-knowledge of design tool capabilities, the design controller's strategic KMs determine which design tool provides the service. The process then proceeds as an explicit linkage.**

By providing such a message based interface to all components, each component (design tool, standards processor, database interface, database management system, user workstation, etc.) becomes a *server* which performs one task. Central coordination and control is provided by message passing through the design controller.[10] This also provides the mechanism for the controller to monitor the actions of the rest of the system. Since database requests will be frequent, and since they require a sophisticated translation, direct communication between any component or database manager and the database interface server is permitted. The communications server structure is depicted in Figure 10.

[10]A distributed control philosophy in which messages are broadcast to all components is another alternative. Each component must be capable of looking at all requests and answering those it feels it can handle, and the requestors must be capable of accepting multiple replies. Some sort of controller or *spy* is needed to insure that each request is handled by someone, and that only the most appropriate tools respond to any request.

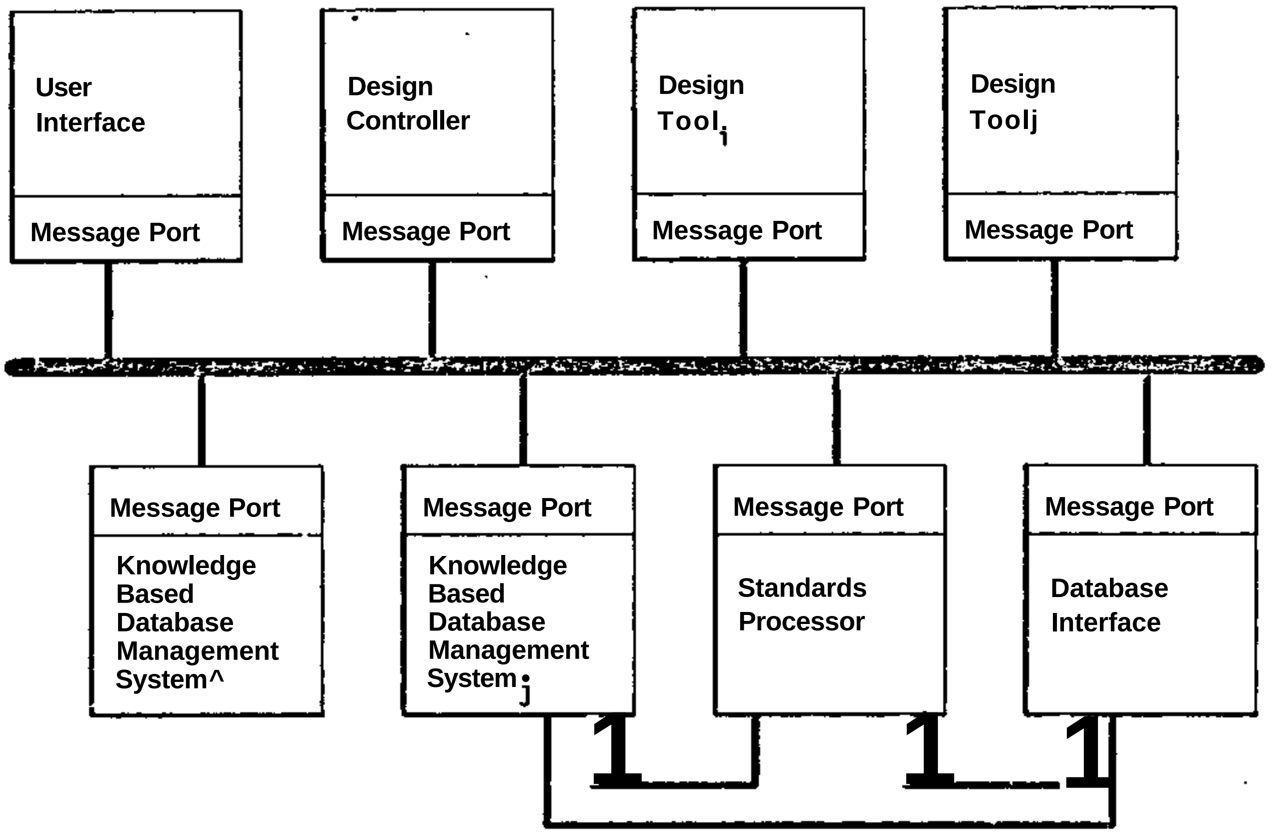


Figure 10: Communications Server Structure

5.7. User Interface

Sophisticated user interfaces are a necessity for ease of use [Hayes 81]. The basic principles of the interface are natural language input, and high resolution graphics output. User interaction occurs on an engineering workstation with a bit-map graphics display, keyboard, and pointing device. Substantial computing resources are dedicated to the interface.

Each workstation provides the capabilities for a single user to interact with the system. The workstations all function as interface servers, passing a user's requests for service to the design controller.

Restricted natural language syntax front ends can be built using available software tools [Hendrix 77, Carbonell 83, Martin 83]. The user should be able to communicate with any tool using the same vocabulary. Some of the principles used for semantic translation of data between subsystems can be applied to the translation of a common input language to the required input format for a tool.

Overlapping multiple window displays and an input pointing device, as exemplified by the *Smalltalk-80* environment [Goldberg 83] are the basis of the output model. Several channels of output can be simultaneously presented to the user.

An example of a first version of such of an interface is graphical output for a system such as *HI-RISE.* Once *HI-RISE* has completed its processing, the best final structural configuration will be displayed.

Any of the other design alternatives can be selected for display. By placing each alternative in its own window, the designer can compare alternatives side by side, or he can examine portions of any design in detail in one window while viewing the entire structure in another window. A simple display of the final design is insufficient; the designer needs continual feedback during the design process. During design, the search tree is displayed and continually updated as design decisions are made, and the *current best design* would be highlighted. The user may redirect the system to consider another design alternative simply by pointing to that part of the tree. An example of such a display is shown in Figure 11.

As advanced I/O devices become available, they will be incorporated into the interface. Two devices which appear most promising are voice input and synthesized speech output. Sketch recognition and digital image input can be anticipated as part of an advanced system.

5.8. Work in Progress

Work is in progress, building components and prototypes of the architecture described above. A variety of hardware and support software is being used to build the system, and is described below. That is followed by a brief discussion of the current status of the system.

5.8.1. Computer Environment

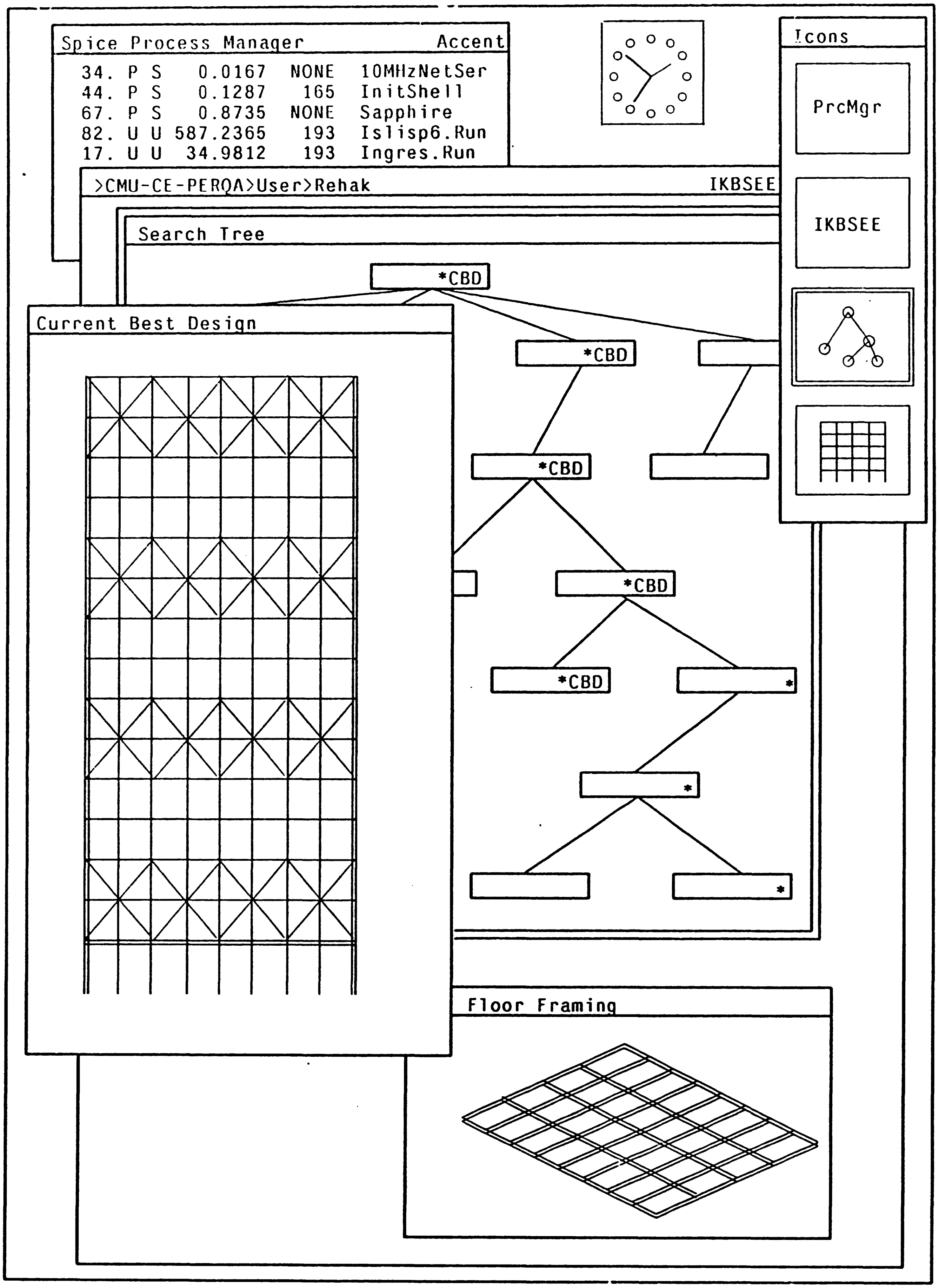
The system is being built from a collection of available hardware and software. The hardware provides the capabilities to build the distributed environment, and a number of distributed computing projects are underway at C-MU. These, in addition to a large body of researchers developing a variety of software tools, permit more rapid development and testing of ideas than could be achieved otherwise.

The user workstations are PERQ personal computers [Newell 79]. The host hardware for the other components is a network of DEC VAX-11/750s and VAX-11/780s. All hardware is connected via Ethernet [Metcalf 76], and file and print servers are included in the network.

The PERQs operates under *ACCENT* [Rashid 81], the VAXes under UNIX [Ritchie 74]. Both operating system provide process, address space, and message passing primitives. Using IPC Qnter-Process Communications) [Rashid 80], individual programs which run in their own address space, can communicate and form an integrated computing environment. Network IPC facilities are available which provide a distributed computing environment without any explicit programming; it is the authors' understanding that the network IPC facilities provided by the PERQ/VAX environment provides capabilities not found in other systems such as APOLLO and SUN workstations. This provides support for the communications facilities described in Section 5.6. Such a network hardware configuration is shown in Figure 12.

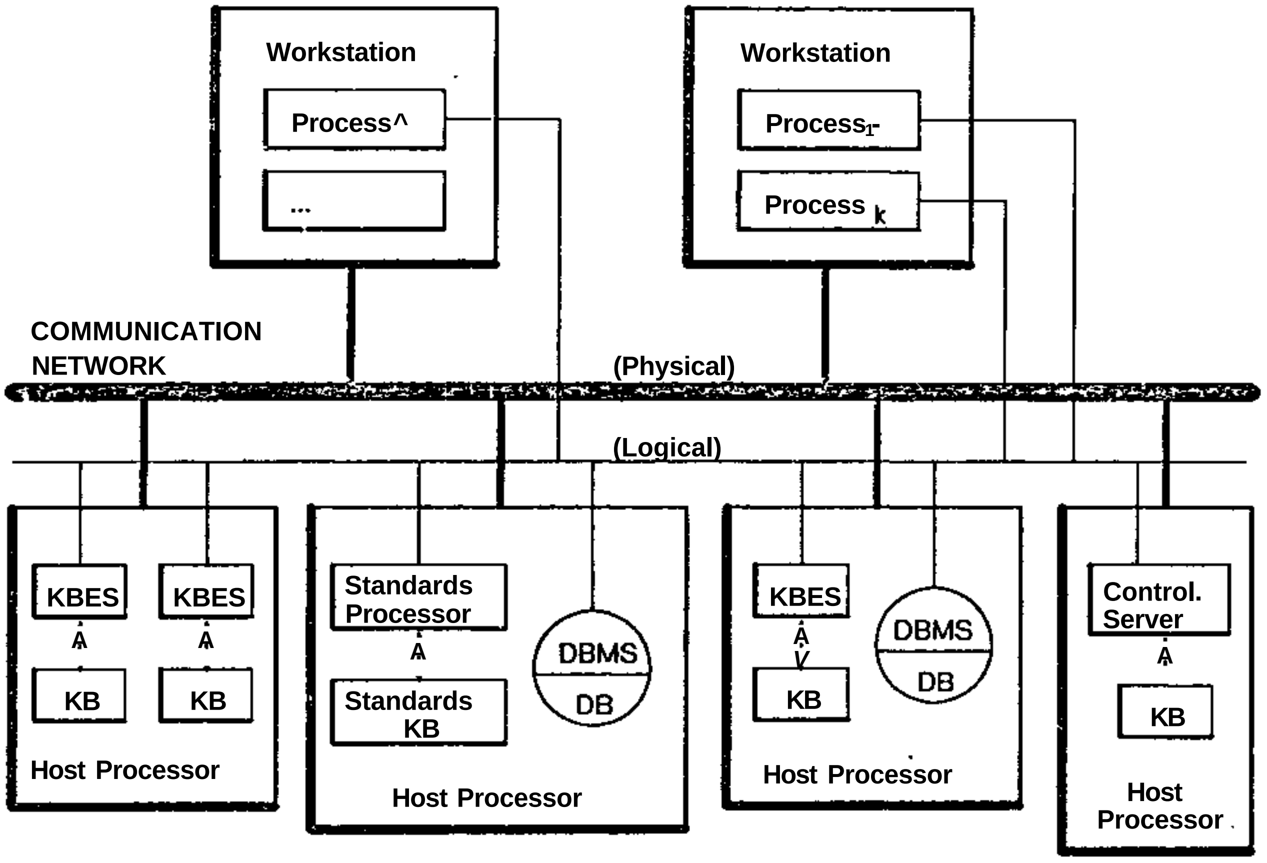
A capability to integrate programs written in different languages is needed; e.g., *FORTRAN* algorithmic tools such as finite element analysis must be combined with *LISP* knowledge based processors. The VAX environment provides the capabilities to integrate *FORTRAN, Pascal, FRANZ LISP* and C, and interlanguage support for *Pascal, COMMON LISP* and C is provided on the PERQs.

A variety of expert system and other development tools are available. *SRL* [Wright 83] is a frame-based language for representing schemas and interschema relations. *PSRL* [Rychener 84a] is a frame-based, OPS-like production system written in *SRL,* in which knowledge is represented through rule sets and schemas (objects) containing slots (attributes) and values. Several versions of the *OPS*



```
34. P S    0.0167  NONE  10MHzNetSer
44. P S    0.1287   165  InitShell
67. P S    0.8735  NONE  Sapphire
82. U U  587.2365   193  Islisp6.Run
17. U U   34.9812   193  Ingres.Run
```

Figure 11: Workstation Display



Figu re 12: Network Environment

family of production systems also are available. Many of these tools are currently being ported to *COMMON LISP* to run on the PERQs. In addition, *INGRES* [Stonebraker 76] and *RIM* [Erickson 81] are available database management systems.

5.8.2. Current Status

Work is concurrently proceeding on a number of projects which form the basis of the entire system architecture described above. At this time, no attempt to build a complete structural engineering environment is underway; only the individual components are being built and tested, and the first steps towards building a distributed environment are being explored.

- The first version of *HI-RISE* has been completed, demonstrating the application of a knowledge based approach to preliminary structural design.

- *DESTINY,* our first blackboard model expert system, is under development and should be operational in 1985.

- A graphical user interface to *HI-RISE* which contains some of the features outlined in Section 5.7 is being completed. The interface permits the alternative designs to be displayed along with the decision context tree as the design process proceeds.

- *SICAD* is operational, but an application program driver is yet to be developed.

- A *LISP* based symbolic decision table interpreter has been developed and is being interfaced with the *OPS* family of expert system framework, providing the basis of an alternative approach to knowledge based standards processing.

- *KADBASE* is being implemented as a system linking two databases, one for structural design and one for cost estimating, with two expert systems. The expert system prototypes are, *HICOST-II*, a revised version of the *HICOST* cost estimator, and *SPOT-CHECKER*, a code compliance spot checker and value engineering system.

- Several small experiments are underway which are aimed at exploring the capabilities and use of the networked IPC environment.

- Since many blackboard model expert system are under development, each using different control strategies, a general purpose, rule-based blackboard system kernel is currently being designed [Rychener 84b]. The kernel system supports multiple tactical KMs which are distributed across many processors, using IPC. The inference mechanisms, focus of attention, etc., used by the kernel are specified through a set of production rules. This kernel will form the basis of the blackboard model for *KADBASE*.

Completing an integrated system would involve a number of steps and development of many design tools, standards, databases, etc. Integrating the tools would then be accomplished by having each component send messages to the controller or other system components, as described in Section 5.6, whenever one tool requires the services of another. Speculation on completing such a system would be premature at this time.

6. Summary

Knowledge based applications will be a key component of the next generation of CAD systems. Work in progress in building knowledge based systems for structural engineering was described, and a variety of issues related to the development of knowledge based expert systems for structural engineering tasks, knowledge based processing of design standards, and database support for knowledge based tasks were presented. The system architecture of an integrated knowledge based structural engineering environment which addresses some of problems areas and which builds on the current research was described.

Work is progressing on building prototype components of the integrated structural engineering environment; *DESTINY*, *SICAD*, *KADBASE*, *HI-RISE* graphics, distributed database servers, and IPC based communications are all subjects of current research. Stepwise development and refinement of such components is the current goal. The long term goal is the creation of an integrated system, and the proposed architecture presents the current view of the master plan of a system which is designed to achieve that goal.

Acknowledgements: The work described above is part of a long term effort in building improved computer based tools for civil engineering applications. The system architecture described above is a result of the cooperative and evolutionary work of several individuals, including S.J. Fenves, M.L. Maher, and L.A. Lopez. In addition, researchers of the C-MU Robotics Institute and Computer Science Department, including M.D. Rychener and M. Fox have provided excellent system building tools and have been very responsive to our needs, continually providing state-of-the-art software. The excellent computer facilities and the ability to interact with numerous individuals developing knowledge based tools and applications in the C-MU environment has had profound impacts on our ability to progress in this research.

7. References

[AISC80] ——————————, *Specification for the Design, Fabrication and Erection of Structural Steel for Buildings,* Eighth Edition, American Institute of Steel **Construction (AISC), Chicago, IL, 1980.**

[Balzer80] **Balzer, R, Erman, L.D. London, P., and Williams, C, "Hearsay-III: A Domain Independent Framework for** Expert Systems," *Proceedings, First Annual National Conference on Artificial Intelligence,* **pp. 108-110, 1980.**

[Bennett 78] **Bennett, J., Creary,** L, **Englemore, R., and Melosh, R.,** *SACON: A Knowledge-Based Consultant for Structural Analysis,* Technical Report STAN-CS-78-699, **Department of Computer Science, Stanford University, September 1978.**

[Blum 82] **Blum, R.L., 'Induction of Causal Relationships from a Time-Oriented Clinical** Database,[11] *Proceedings, Second National Conference on Artificial Intelligence,* **pp. 355-357, 1982.**

[Carbonell 83] **Carbonell, J., Boggs, M., Mauldin, M., and Anick, P., "The XCALIBUR Project: A Natural Language Interface to Expert Systems,"** *Proceedings, Eighth* ***International Joint Conference on Artificial*** *Intelligence,* **pp. 653-656, August 1983.**

[Cardenas 80] **Cardenas. A.F., and Pirahesh, M.H., "Data Base Communications in a Heterogeneous Data Base Management System Network,"** *Information Systems,* **Vol.5, No. 1, pp. 55-79, 1980.**

[Christman 83] **Christman, A.M., "The Coupling of CAD and CAM,"** *Computers in Mechanical Engineering (CIME),* **American Society of Mechanical Engineers (ASME), Vol. 2, No. 2, pp. 26-29, September 1983.**

[Eastman 81] **Eastman, CM., "Database Facilities for Engineering Design,"** ***Proceedings of the IEEE$_t$*** **Institute for Electrical and Electronics Engineers (IEEE), Vol. 69, No. 10, pp. 1249-1263, October 1981.**

[Erickson 81] **Erickson, W.J., Gray, F.P., and Limbach, G.,** *Relational Information Management System,* **Version 5.0, Boeing Commercial Airplane Co., Seattle, WA, 1981.**

[Erman 80] **Erman, L.D., Hayes-Roth, F., Lesser, V.R., and Reddy, D.R., "The Hearsay-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty,"** ***Computing Surveys$_t$*** **Association for Computing Machinery (ACM), Vol.12, No. 2, pp. 213-253, February 1980.**

[Fenves 73] **Fenves, S.J., "Representation of the Computer-Aided Design Process by a Network of Decision Tables,[11]** ***Computers and Structures,*** **Vol. 3, No. 5, pp. 1099-1107, September 1973.**

[Fenves 79a] **Fenves, S.J.,** ***Functional Specifications for Standards Processing Software,*** **Technical Report R-79H, Department of Civil Engineering, Carnegie-Mellon University, Pittsburgh, PA, April 1979.**

[Fenves 79b] **Fenves, S.J.,** ***Functional Specifications for Standards Processing Software,*** **Technical Report R-79-12, Department of Civil Engineering, Carnegie-Mellon University, Pittsburgh, PA, June 1979.**

[Fenves 81] Fenves, S.J., "Computer-Aided Design in Civil Engineering," *Proceedings of the IEEE,* VoJ. 69, No. 10, pp. 1240-1248, October 1981.

[Fenves 82a] Fenves, S.J., and Rasdorf, W.J., *Treatment of Engineering Design Constraints in a Relational Database,* Technical Report DRC-12-14-82, Design Research Center, Carnegie-Mellon University, Pittsburgh, PA, December 1982.

[Fenves 82b] Fenves, S.J., and Rasdorf, W.J., "Role of Database Management Systems in Structural Engineering," *Proceedings, IABSE Workshop on Informatics in Structural Engineering,* Bergamo Italy, International Association of Bridge and Structural Engineers (IABSE), Zurich, pp. 229-242, October 1982.

[Fjellheim 83] Fjellheim, R., and Syversen, P., *An Expert System for Sesam-69 Structural Analysis Program Selection,* Technical Report CP-83-6010, Division for Data Technology, Computas, Norway, January 1983.

[Foderaro 82] Foderaro, J.K., and Sklower, K.L., *The FRANZ LISP Manual,* University of California at Berkeley, 1982.

[Forgy81] Forgy, C.L., *OPS5 User's Manual,* **Technical Report** CMU-CS-81-135, **Department of Computer** Science, **Carnegie-Mellon University, Pittsburgh, PA, July 1981.**

[Genesys 76] Genesys Limited, *GENESYS,* Technical Report, Genesys Limited, Loughborough, England, 1976.

[Gero83] Gero, J.S., *Knowledge Engineering - Future of Computers in Engineering,* unpublished presention at the Institution of Engineers, Australia, 1983.

[Goel71] Goel, S.K., and Fenves, S.J., "Computer-Aided Processing of Design Specifications," *Journal of the Structural Division,* American Society of Civil Engineers (ASCE), Vol. 97, No. ST1, pp. 463-479, January 1971.

[Goldberg 83] Goldberg, A., and Robson, D., *Smalltalk-80. The Language and its Implementation,* Addison-Wesley, Reading, MA, 1983.

[Hayes 81] Hayes, P.J., Ball, E., and Reddy, R., "Breaking the Man-Machine Communication Barrier," *Computer,* IEEE Computer Society, Vol. 14, No. 3, pp. 19-30, March 1981.

[Hendrix 77] Hendrix, G.G., *The LIFER Manual: A Guide to Building Practical Natural Language Interfaces,* SRI **International, 1977.**

[Holtz82] **Holtz, N.M.,** *Symbolic Manipulation of Design Constraints. An Aid to Consistency Management,* **Technical Report DRC-02-12-82, Design Research Center, Carnegie-Mellon University, Pittsburgh, PA, April 1982.**

[Howard 83] **Howard, H.C.,** *HICOST,* **unpublished Project Report, 12-743 Expert Systems in Civil Engineering, Department of Civil Engineering, Carnegie-Mellon University, Pittsburgh, PA, December 1983.**

[Howard 84] **Howard, H.C.,** *Integrating Knowledge-Based Systems with Database Management Systems for Structural Engineering Applications,* unpublished Ph.D. Thesis **Proposal, Department of Civil Engineering, Carnegie-Mellon University, Pittsburgh, PA, April 1984.**

[Kellogg 83] Kellogg, C. H.$_t$ "Intelligent Assistants for Knowledge and Information Resources Management," *Proceedings, Eighth International Joint Conference on Artificial Intelligence,* 1983.

[Kemighan 78] Kernighan, B.W., and Ritchie, D.M., *The C Programming Language,* Prentice-Hall Inc., Englewood Cliffs, NJ, 1978.

[Lopez 72] Lopez, L.A., "POLO: Problem Oriented Language Organizer," *Computers and Structures,* Vol. 2, No. 4, pp. 555-572, September 1972.

[Lopez 77] Lopez, L.A., "FINITE: An Approach to Structural Mechanics Systems,'^{1} *International Journal for Numerical Methods in Engineering,* Vol. 11, pp. 851-866, 1977.

[Lopez 84a] Lopez, L.A., Elam, S.L., and Christopherson, T., "SICAD: A Prototype Implementation System for CAD," *Proceedings, ASCE Third Conference on Computing in Civil Engineering,* San Diego, CA, American Society of Civil Engineers (ASCE), pp. 84-94, April 1984.

[Lopez 84b] Lopez, L.A., and Elam, S.L., *SICAD: A Prototype Knowledge Based System for Conformance Checking and Design,* unpublished article, 1984.

[Maherfr*] Maher, M.L., and Fenves, S.J., "HI-RISE: An Expert System for the Preliminary Structural Design of High Rise Buildings," *Knowledge Engineering in Computer-Aided Design,* IFIP Working Group 5.2 Working Conference, Budapest, Hungary, International Federation for Information Processing (IFIP), September 1984.

[Martin 83] Martin, P., Appelt, D., and Pereira, F., "Transportability and Generality in a Natural-Language Interface System,^{11} *Proceedings, Eight International Joint Conference on Artificial Intelligence,* pp. 573-581,1983.

[Metcalf76] Metcalf, R., and Boggs, D., "Ethernet: Distributed Packet Switching for Local Computer Networks," *Communications of the Association for Computing Machinery (CACM),* Association for Computing Machinery (ACM), Vol. 19, No. 7, pp. 395-404, July 1976.

[Newell 79] Newell, A., Fahlman, S., and Sproull, R.F., *Proposal for a Joint Effort in Personal Scientific Computing,* Technical Report, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, August 1979.

[Nii 82] Nii, P., Feigenbaum, E.$_f$ Anton, J., and Rockmore, A., "Signal-to-Symbol Transformation: HASP/SIAP Case Study," *AI Magazine,* Vol. 3, No. 2, pp. 23-35, Spring 1982.

[Preiss83] Preiss, K., "Future CAD Systems," *Computer-Aided Design,* Vol.15, No. 4, pp. 223-227, July 1983.

[Rashid 80] Rashid, R., *A Proposed DARPA Standard Inter-Process Communication Facility for UNIX Version Seven,* Technical Report, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, February 1980.

[Rashid81] Rashid, R., and Robertson, G.G., *Accent: A Communication Oriented Network Operating System Kernel,* Technical Report CMU-CS-81-123, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, April 1981.

[Rehak81] Rehak, D.R., and Lopez, L.A., *Computer Aided Engineering: Problems and Prospects,* Civil Engineering Systems Laboratory Research Series (CESLRS) 8, Department of Civil Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, July 1981.

[Ritchie 74] Ritchie, D.M., and Thompson, K., "The UNIX Time-Sharing System,'1 *Communications of the Association for Computing Machinery (CACM),* Association for Computing Machinery (ACM), Vol. 17, No. 7, pp. 365-375, July 1974.

[Rivlin 80] Rivlin, J.M., Hsu, M.B., and Marcal, P.V., *Knowledge-Based Consultation for Finite Element Analysis,* Technical Report AFWAL-TR-80-3069, Flight Dynamics Laboratory (FIBRA), Wright-Patterson Airforce Base, Dayton, OH, May 1980.

[Roos 66] Roos, D., *ICES System Design,* MIT Press, Cambridge, MA, 1966.

[Rychener 84a] Rychener, M.D., and Fox, M., *PSRL: An SRL-Based Production System. Reference Manual for PSRL Version 1.2,* Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA, 1984.

[Rychener 84b] Rychener, M.D., *A Rule Based Blackboard System Kernel,* Technical Report, Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA, 1984, forthcoming.

[Shortliffe 76] Shortliffe E.H., *Computer-Based Medical Consultations: MYCIN,* American Elsevier, New York, 1976.

[Sriram81] Sriram, D., and Motazed, B., *SPECON - A Specification Consultant,* unpublished Project Report, Department of Civil Engineering, Carnegie-Mellon University, Pittsburgh, PA, 1981.

[Sriram 83] Sriram, D., *Knowledge-Based Approach to Integrated Structural Design,* unpublished Ph.D. Thesis Proposal, Department of Civil Engineering, Carnegie-Mellon University, Pittsburgh, PA, October 1983.

[Sriram 84] Sriram, D., Maher, M.L., and Fenves, S.J., "Knowledge-Based Expert Systems in Structural Design," *Proceedings, NASA Conference on Advances in Structural Mechanics,* Washington, DC, October 1984, in press.

[Stahl 83a] Stahl, F.I., *The Standards Interface for Computer-Aided Design,* Technical Report NBSIR 83-2671, Center for Building Technology, National Bureau of Standards, Washington, DC, April 1983.

[Stahl 83b] Stahl, F.I., Wright, R.N, Fenves, S.J., and Harris, J.R., "Expressing Standards for Computer-Aided Building Design,'1 *Computer-Aided Design,* Vol.15, No. 6, pp. 329-334, November 1983.

[Stirk81] Stirk, J., *Two Software Aids for Design Specification Use,* unpublished Master's Thesis, Department of Civil Engineering, Carnegie-Mellon University, Pittsburgh, PA, October 1981.

[Stonebraker 76] Stonebraker, M., Wong, E., and Kreps, P., "The Design and Implementation of INGRES,1' *Transactions on Database Systems (TODS),* Association for Computing Machinery (ACM), Vol. 1, No. 3, pp. 189-222, September 1976.

[vanMelleSi] van Melle, W., Shortliffe, E.H., and Buchanan, B.G., "EMYCIN: A Domain-Independent System that Aids in Constructing Knowledge-Based Consultation Programs," in *Machine Intelligence,* Vol. 9, *Infotech State of the Art Report,* No. 3, Infotech, 1981.

[Wright 83] Wright, M., and Fox, M., *SRU Schema Representation Language,* Technical Report, Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA, December 1983.

Table of Contents

List of Figures

List of Tables