Role of Artificial Intelligence and Knowledge-Base Expert System Methods In Civil Engineering

by

S. Fenves

EDRC-12-17-87

Role of Artificial Intelligence and Knowledge-Base Expert System Methods in Civil Engineering[1]

Steven J. Fenves[2]

ABSTRACT

Present use of computers in civil engineering is largely devoted to numeric, algorithmic calculations. This mode is not appropriate for the empirical, heuristic, ill-structured problems of civil engineering practice. The paper reviews recent work in Artificial Intelligence and Expert systems addressing these latter issues, identifies the distinctive features of engineering knowledge based systems, the roles of such systems, and attempts to predict their evolution.

1. Introduction

The first published reference on the use of a digital computer in civil engineering that I have found dates to 1952 ([Bennett 52] quoted in [Livesley 66]). In the succeeding 35 years, computer use has undergone at least three "revolutions":

- High-level procedural languages such as *FORTRAN, ALGOL,* etc., vastly increased the number of engineers who could communicate their problem directly to the computer (i.e., the compiler), without having to rely on a coder to translate the problem into machine instructions;
- Time-shared systems provided a *rate* of man-machine interaction commensurate with the engineer's needs; when coupled with higher-level problem-oriented languages, they also vastly increased the *level* of the man-machine dialogue; and
- Personal computers and workstations have placed the *control* of the man-machine problem-solving process in the hands of the engineer.

As a result of these revolutions, the computer has become an integral and indispensable ingredient of civil engineering practice, research and education. The computer has also vastly accelerated the

[1] Prepared for presentation at Ohsaki Research Institute Fifth Anniversary Seminar, Tokyo, Japan, July 21-23,1987

[2]Sun Company University Professor of Civil Engineering, Carnegie Mellon University, Pittsburgh, PA, U.S.A.

*technology transfer*between research and practice and between industries, as well as entire economies, at different levels of development. A methodology embodied in a computer program enjoys orders of magnitude faster and wider dissemination than its description in an engineering journal. Similarly, the finite element method is a salient example of rapid transference from the "high-tech" aerospace industry to "low-tech" industries such as civil engineering or shipbuilding.

Notwithstanding the enormous volume of computer use in civil engineering, this use today is largely concentrated in one category, namely *calculating:* the mapping of one set of numbers, representing the problem at hand, to another set of numbers, representing the results or outcome, according to a predefined procedure, called the *algorithm.* Thus, the computer is primarily involved in the derivation of *predicted* consequences of actual or proposed engineering decisions.

Computer use is also being rapidly extended into two other categories: *presenting* information in graphic, textual and other forms; and *sharing* information among individuals and organizations participating in a common project or enterprise. The lively interest in interactive computer graphics, the rapidly growing use of CADD, and the increasing integration of text processing into all phases of engineering are manifestations of the first category, while the growing emphasis on engineering database management systems (DBMS) as a critical ingredient of computer-integrated manufacturing (CIM) and computer-integrated construction (CIC) is a salient example of the latter.

By contrast to the above explosive developments, computer use has been seriously lagging in a fourth category, that of *reasoning.* Reasoning differs from computing in two key aspects. First, the objects dealt with, or more precisely, the *representations* of these objects, are symbolic rather than numeric, whether these objects are symbolic relations, geometric entities or conceptual entities. Second, the processes operating on these objects are *ill-structured,* involving assumptions, approximations, "rules of thumb" and other *heuristics* [Simon 81]. The processes that are most characteristic of engineering, notably design and planning, fall into this category. Computer aids for this category have lagged considerably behind computational aids.

The above does not imply that civil engineers have not attempted to develop programs for design and planning, but rather that the present programs are not based on an intellectual framework that explicitly deals with symbolic operands and heuristic operations. Design programs do exist, but they have been developed in an algorithmic framework, with assumptions and heuristics deeply buried in masses of procedural code. Such programs tend to be highly restrictive in their scope and highly *opaque,* in that they function as "black boxes" with no mechanisms to explain the heuristic bases underlying their processes. These programs are also notoriously difficult to understand, update or modify. As a typical example, one may find a program for the design of reinforced concrete columns that computes stirrup spacing as the minimum of the spacing needed for shear reinforcement or 12 inches. If one digs deep enough, one finds that the 12-inch limit was introduced thirty years ago when prefabrication of reinforcing cages was not common, and ironworkers needed that spacing so as to use previously tied stirrups as a ladder for climbing. The explanation or justification of this heuristic is not part of the program, and the heuristic is buried so deeply among code-mandated requirements that it has remained in the program ever since.[3]

There is a pent-up frustration, among program users and developers alike, with the limitations and shortcomings of present-day algorithmic, numeric programs. Users are frustrated because the programs implement someone else's heuristics without explanations or ways of substituting their own, while programmers are frustrated because they don't have tools for implementing what the users want. Hence the growing interest in a new program development methodology which has grown out of research in Artificial Intelligence.

2. Artificial Intelligence and Knowledge-Based Expert Systems

The discipline of Computer Science has grown up in parallel with the growth and proliferation of computers. Computer users in general, and civil engineers in particular, have benefited from research results from such branches of Computer Science as numerical methods, language theory and programming systems. Particularly significant impacts on the manner in which civil engineering programs are designed, developed, used and maintained have come from the discipline of *software engineering,* a significant "spinofT from Computer Science.

As early as the 1950's, a branch of Computer Science began to explore symbolic, as opposed to numeric, processing. This branch evolved into Artificial Intelligence (AI), concerned with the dual issues of constructing computer programs that appear to be intelligent and of understanding human intelligence, or human problem solving, by means of programs that emulate humans. AI today is a mature science, dealing with intelligent systems, search methodologies, knowledge representation, vision, natural

. ^e author is aware of this heuristic only because he programmed it in 1957.

language processing and robotics. AI research on the representation and processing of knowledge over the past thirty years has recently produced a "spinoff, comparable in its potential impact to software engineering, called *knowledge engineering,* concerned with the development of programs variously referred to as knowledge-based systems, expert systems, or knowledge-based expert systems (KBES). KBES have a particularly high potential for practical use in ill-structured domains where knowledge is highly heuristic and explicit algorithms either don't exist or provide only limited and restricted problem-solving capabilities. Thus, KBES *appear to* provide exactly the type of conceptual framework that is needed for the design and planning applications in civil engineering which have been unsuccessfully attempted by algorithmic means. The purpose of this paper is to explore how this appearance can be converted into reality.

Definition of KBES. A good standard definition of KBES is the following:

"Knowledge-based expert systems are interactive computer programs incorporating judgment, experience, rules of thumb, intuition, and other expertise to provide knowledgeable advice about a variety of tasks [GaschingBI]."

The first reaction of many professionals active in computer-aided engineering to the above definition is one of boredom and impatience. After all, conventional computer programs for engineering applications have become increasingly interactive; they have always incorporated expertise in the form of limitations, assumptions and approximations^ discussed above; and their output has long ago been accepted as advice, not as "the answer" to the problem.

There is a need, therefore, to add an operational definition to distinguish the new wave of KBES from conventional algorithmic programs which incorporate substantial amounts of heuristics about a particular application area, or *domain.* The distinction should not be based on implementation languages, e.g., *FORTRAN vs. LISP* (after all, KBES frameworks are now available in *Pascal or C* implementations), or any absolute separation between domain-dependent knowlege base and generic inference engine (for example, in frame-based knowledge representations there is no generic inference engine.)

The principal distinction between KBES and algorithmic programs lies in the use of knowledge. A traditional algorithmic application is organized into two parts: data and program. An expert system separates the program into an explicit *knowledge base* describing the problem solving knowledge and a control program or *inference engine* which manipulates the knowledge base. The data portion or *context* describes the problem being solved and the current state of the solution process. Such an approach is denoted as *knowledge-based* [Nau 83].

Architecture of KBES. A variety of KBES architectures are available. Various KBES frameworks have different inference procedures and different knowledge representation schemes, including: production systems [Forgy 81], semantic inference networks [Reboh 81], and frame representations [Wright 83]. More complex blackboard systems, based on multiple experts operating at different levels of abstraction, have also been built [Balzer 80, Erman 80, Nii 82]. In the production system formalism for domain knowledge representation, the knowledge is represented directly in terms of IF-THEN rules. Some of the problem solving strategies incorporated in KBES are discussed in [Maher 86].

The basic components of the KBES using the production system (IF-THEN rules) formalism are:

- *Knowledge Base.* The knowledge base is the repository for the domain knowledge used by the system in the form of rules. The knowledge base may also contain long term historical information and *facts.*
- *Context* The context contains all of the information which describes the problem currently being solved, including both problem data and solution status. The problem data may be divided into facts provided by the user and those derived or inferred by the program. A session with an KBES begins with the user entering some known facts about the problem into the context.
- *Inference Engine.* The inference engine operates on the context, utilizing the rules in the knowledge base to deduce new facts which then can be used for subsequent inferences. The basic operation of a *forward chaining* inference engine is an infinite loop performing three steps:
    1. Examine the premises of rules in the knowledge base and determine which of these currently evaluate to *true,* based on the current problem data maintained in the context. This step, performed by the change monitor or pattern matcher, yields a set of candidate rules.
    2. Select one of the applicable rules. The rule is chosen by the scheduler or conflict resolver.
    3. Invoke or *fire* the corresponding action, which will change some data items in the context. The context is updated by the knowledge modifier. As a result of step 3., other rules may become candidates to fire on the next cycle.

The objective of the inference engine is to arrive at a global conclusion *(goat),* and the process continues until the context is transformed into the desired goal state, or when there are no more rules remaining to be fired. It is to be emphasized that only the knowledge base of a KBES is domain specific. All the other components are parts of a general purpose KBES framework applicable to a range of application domains.

Many KBES inference engines can also deal with imprecise, inexact or incomplete knowledge.

Associated with the data are certainty measures indicating the level of confidence in the data. Rules can conditionally fire **based** on the certainty of their premise, and can have certainty factors associated with their conclusion. The inference mechanism can then propagate certainty about the inferences along with results of the inferences.

In addition to the three basic components, it is highly desirable that the KBES contain three additional components:

- *Explanation Module.* The explanation module provides the KBES with the capability to explain its reasoning and problem solving strategy to the user. At any point the user may interrupt the system and inquire *why* it is pursuing the current line of reasoning. In addition, the program can explain *how any* conclusion was deduced and how knowledge was applied.
- *Knowledge Acquisition Module.* The information in the knowledge base is in a rigid format, and the translation of knowledge obtained from experts to the required internal format may be tedious. The knowledge acquisition module aids in this task. Although it is desired that eventually the domain expert be able to enter directly knowledge into the system, this goal is currently not achieved.
- *User Interface.* The user accesses the system through a friendly interface, often using a domain oriented subset of a natural language, menus or computer graphics. The interface provides capabilities for the user to monitor the performance of the system, volunteer information, request explanations, and redirect the problem solving approach.

The basic concepts of a knowledge base, knowledge acquisition, explanation, context and inference mechanisms are common to the different types of KBES architectures. Details of system organization, knowledge and data representation, and inference method vary among the different approaches.

KBES Application Areas. KBES applications are appearing in many disciplines. However, not all problem-solving tasks are amenable to KBES formulation. The following is a partial list of criteria for the evaluation of promising potential applications [Hayes-Roth 83]:

- There are recognized experts in the field whose performance is better than that of novices.
- The factual component of domain knowledge is routinely taught to neophytes who become experts by developing their own rules and empirical associations.
- Typical tasks are performed by an expert in a few minutes to several hours.
- Tasks are primarily cognitive, requiring reasoning at multiple levels of abstraction.
- Algorithmic solutions are either impractical or result in overly constrained or specialized programs.
- There are substantial benefits in applying the expert knowledge to each occurrence of the task.

A practical KBES should have the following characteristics:

- *Usefulness.* The KBES must be capable of performing useful functions. Usefulness depends on the domain and task for which the KBES is developed.

- *Performance.* The KBES must have a high level of performance, reliability and accuracy over a range of application cases. This requires that the expert system have the specialized knowledge that separates human experts from novices.
- *Transparency.* An KBES is transparent if it can be understood by people using it. To have this characteristic, the KBES must be able to explain its actions and reasoning to the user.

The range of potential KBES applications covers a spectrum from *derivation* or *interpretative* problems to *formation* or *generative* problems [Amarel 78]. In derivation problems, the problem conditions and description are given as part of a solution description (goal). The KBES completes the solution description by applying the available knowledge and rules such that the initial data and conditions are well integrated in the solution. As an example, in a derivation problem such as theorem proving, a solution hypothesis is formulated which the expert system attempts to prove by applying rules to the known data. Repeated application of rules transforms the problem statement (the initial state) to the solution state. By contrast, in formation problems conditions are given in the form of (constraints) that the solution as a whole must satisfy. Candidate solutions are generated and tested against the specified constraints. Two subclasses are: *constraint satisfaction* in which the solution need only satisfy the governing constraints, and *optimization* where an attempt is made to find the optimal solution. The design of a plan, object, or system fits this paradigm. Most actual problems are neither pure formation nor derivation problems, but lie somewhere in-between and require that techniques from both categories be used in problem solving.

The following list of problem types covers the spectrum of KBES applications.

- *Interpretation.* An interpretation system takes observed data and explains its meaning by inferring the problem state which corresponds to the observed data. Examples are *Dipmeter Advisor,* a system for interpreting geophysical oil well log data [Davis 81, Smith 83], and *Prospector,* a system for identifying geological ore-bearing formations [Duda 79].
- *Diagnosis.* Diagnosis systems infer malfunctions or system state from observed irregularities and interpretation of data. *MYCIN,* an infections disease diagnostician [Shortliffe 76], and several other medical diagnosis programs fall into this category.
- *Monitoring.* A monitor observes system behavior and compares the observations to the planned behavior to determine flaws in the plan or potential malfunctions of the system. An example is *Ventilation Manager,* a program for monitoring a patient's ventilation therapy [Fagan 79].
- *Design.* Design is the process of developing a configuration for an object which satisfies all applicable constraints. *R1* (or *XCON)* is an example of a design system which is used to configure VAX[4] computers [McDermott 80].
- *Planning.* Planning is a design process that yields a set a actions intended to produce a

[4]VAX is a registered trademark of Digital Electronics Corp.

desired outcome. An example is *MOLGEN,* an KBES for planning experiments in molecular genetics [Stefik 81].

- *Control.* A control system encompasses many of the characteristics of the other types of applications described above. It must interpret data, predict outcomes, formulate plans, execute the plans, and monitor their execution.

Interpretation, diagnosis and monitoring lie at the derivation end of the spectrum while design, planning, and control lie at the formation end.

The Knowledge Engineering Process. Knowledge engineering is an incremental cooperative process currently performed by two sets of people. The first is the *domain expert* who possesses the problem solving knowledge for the problem area being addressed. The second is the *knowledge engineer* who gathers expertise from the domain expert and translates this knowledge into the format required by the expert system framework. A knowledge engineer who is also literate in the application domain is desired, as he can understand the issues involved and the nomenclature used by the domain expert [Dym 84]. As will be discussed later, it is expected that in the near future the knowledge engineers will increasingly be application programmers in the domain.

The knowledge engineering process of building an KBES application is outlined below [Reboh 81, Hayes-Roth 83]. This process is similar in nature to building an algorithmic program or producing the design of an object.

- *Problem Identification.* The first step is to identify the problem to be solved and the characteristics of the solution. Identification of resources, domain experts, and computer facilities are made and overall goals for the project are set.
- *System Design.* The overall structure of the system is selected. Based on processes used by the expert, available data, strategies, information flow, etc., a preliminary model of problem solving to be used by the system is developed. From this, a problem solving strategy and a candidate domain independent KBES development framework or *shell* is selected and key concepts are formalized. If the selected system appears to have the correct formalisms for problem solving and knowledge representation, a more detailed analysis is undertaken to produce a detailed design of the system.
- *Knowledge Acquisition.* Knowledge acquisition is the process of gathering the expert knowledge from the domain expert. This process may be difficult. In some instances the cognitive portions of the problem solving process used by the expert have never been verbalized and are difficult to extract (the expert is not conscious of how he solves problems). In other cases the expert may feel threatened by a computerized replacement and will be reluctant to cooperate.
- *Implementation.* The knowledge engineer's tasks is then implementation; the expert's knowledge is encoded in the format required by the KBES framework. This yields a prototype operational program.
- *Testing.* Once a prototype or partial system is developed, it is tested. The domain expert and the program are given the same problem and their problem solving behavior is

compared. Flaws in the system are detected and corrected by adding or modifying the knowledge used by the program.

The knowledge engineering process is not sequential, and a number of iterations on the last three steps are always necessary to expand, correct and tune the system's behavior. Adding depth of knowledge, breadth of capabilities, and improved interfaces and explanations to the prototype yields the final version of the system.

3. Distinction of Engineering KBES

KBES are being developed for a wide range of civil engineering applications. A recent survey discusses KBES applications grouped under the categories of construction, structural, geotechnical, environmental and transportation engineering [Kim 86]. The majority of the KBES surveyed fall into the category of interpretation and diagnosis, but there are some planning and design KBES surveyed.

The trends in civil engineering, as well as related work in other engineering disciplines, are beginning to highlight the distinctive characteristics of engineering KBES, which may not be present in KBES addressing other domains. The following is an attempt to identify these characteristics.

Use of Causal Knowledge. In some of the KBES literature, distinction is made between "shallow" vs. "deep" knowledge. These terms are misleading and imprecise. A more useful distinction is between *empirical associations* versus *causal knowledge.* The first term includes heuristics observed to be useful or practical, but without any underlying knowledge of the causality between the premise and the conclusion or inference: "IF the load is too large THEN the structure will fail" is an empirical association known since the Stone Age, but today's engineers have extensive causal knowledge relating loads to failure modes. Furthermore, an experienced structural engineer possesses a great deal of *compiled knowledge,* and can confidently make associations between loads and potential failure modes without having to resort to textbook knowledge or basic principles about stresses, strains, distortions, etc. Thus, while it is conceivable that successful KBES can be built in some domains based exclusively on empirical associations, it is almost a foregone conclusion that every engineering KBES must be based, at least in part, on compiled, causal knowledge. Appropriate general mechanisms for explicitly representing such knowledge are still lacking, and today a KBES developer may be forced to disguise such knowledge as purely empirical: a rule of the form "IF compression element is slender THEN instability is a highly likely failure mode" does not explicitly convey its causal source.

Interaction With Algorithmic Components. As indicated above, engineering problem-solving relies extensively on causal knowledge. Furthermore, much of this causal knowledge is either already embodied in existing algorithmic programs, or can be effectively incorporated into *procedural attachments* to rules in a KBES. For example, to establish the premise of the rule "IF every bar in the truss is incorporated in a triangle THEN truss is geometricaly stable", many further IF-THEN rules must be tested, and even then the rule will lack generality (i.e., it will not handle compound trusses). A much more elegant and general rule would be "IF determinant of equilibrium equations is greater than zero THEN truss is geometricaly stable", with the premise established by a procedural attachment which uses a standard matrix procedure. Thus, in any engineering KBES, interaction with algorithmic components is an absolute necessity.

Two levels of interaction are possible. At one level, the KBES may be *interfaced*with an algorithmic program, acting essentially as an intelligent front-end, assisting the user to prepare input data to and interpret results from the algorithmic program, the latter still serving as a monolithic "black box". A typical interaction of this kind is described in [Zumsteg 85]. Alternately, the knowledge-based and computational components may be tightly *integrated,* with knowledge-based modules paired with the functional modules of the algorithm, with each knowledge-based module providing advice, checks, shortcuts, selections, etc. for its corresponding functional module.

The early KBES environments, geared to derivation or diagnostic applications based on "shallow", empirical knowledge, did not provide convenient facilities for interaction with algorithmic components. If interaction was possible at all, it had to be performed by means of ad-hoc arrangements, usually at the operating system level. The more recent KBES environments provide much more convenient interfacing capabilities, either at the level of the knowledge representation languages, as in OPS83 [Forgy 84], or at the level of the control structure of the inference engine.

Interaction With Databases. A common characteristic of engineering problem-solving activities is their intense use of data, in the form of reference information, information on past projects relevant to the current project, shared common information among participants in the project, and information generated by the individual project participants. Increasingly, all such information is stored in, retrieved from and managed by engineering design database management systems (DBMS). A major engineering KBES cannot restrict itself to using only the local context provided by the KBES environment, and has to interact with the information residing in databases external to the KBES.

The two levels of interaction between KBES and databases needed roughly parallel the levels of interaction with algorithmic components discussed previously. At one level, the KBES acts as an intelligent interface **between** the user and the database, assisting in formulating queries to the DBMS. For example, the **KBES** may contain a high-level "rule" of the form "IF there is a previous project similar to the present one THEN use its parameters as the initial values for the current project[1]*. Such a KBES would need many further heuristic rules on what constitutes a "similar project, what the relevant parameters are, etc. so as to formulate a DBMS query to a database of past projects and to retrieve from the query results the parameter needed. In the opposite scenario, the DBMS managing an integrated project database may have a database semantic consistency rule of the form "IF structural component dimensions compatible with architectural requirements AND capacity adequate for imposed mechanical loads THEN accept ELSE reject the database update". In this case, the DBMS would have to execute an appropriate KBES in order to evaluate the consistency rule (as well as to notify the participants involved of the reasons for rejecting any transaction).

Some KBES in the first category have been described in the literature. An experimental prototype interface between multiple KBES and DBMS, *KADBASE,* has been developed by Howard and Rehak [Howard 86]. *KADBASE* provides a flexible interface in which multiple KBES and multiple design databases communicate as independent, self-descriptive components within an integrated CAD system operating in a distributed computing environment. *KADBASE* provides local syntactic and semantic translations of transactions to and from a global data definition. Thus, a KBES issues a query in its own language. *KADBASE* translates the query to the global model; ascertains which database(s) contain the data sought; translates the global description to the local language of the DBMS; and then causes the DBMS execute the query. The query results are similarly translated back to the language of the querying KBES. An example of a KBES to DMBS interface using *KADBASE* is presented in [Garrett 86].

Geometric Reasoning. **A** final distinguishing characteristic of most engineering KBES, particularly those in civil engineering, is their extensive use of spatial attributes of objects (e.g., their dimensions and locations) and of spatial relations among the objects (e.g., connected, adjacent, above, accessible, etc). It may be argued that spatial attributes and relations constitute the *syntax of* any language for reasoning about engineering objects, and that the functional attributes and relations among these objects constitute the *semantics* of that language [Baker 87]. Furthermore, in most engineering problems, objects have multiple functions, and thus multiple semantics. For example, in building design, a single object such as a wall has distinct structural, architectural, acoustic, etc. functional attributes. Design decisions based on one functional domain of an object affect its functional performance in all other domains.

A newly emerging field dealing with the representation and processing of spatial attributes is designated as *geometric reasoning.* The objective of geometrical reasoning is to develop appropriate representations, and operations on those representations, that can support a variety of functional domains. Geometric reasoning can relieve the knowledge engineer concerned with a particular functional domain from having to provide detailed implementations of relations such as "above" or "connected", thereby being able to concentrate on the functional domain. There are at present no general-purpose geometric reasoning systems that can be directly incorporated into engineering KBES. However, due to the prevalence of geometric reasoning needs in many disciplines, notably robotics, it can be expected that such systems will rapidly emerge.

4. Role of Engineering KBES

The present and expected future KBES applications can be roughly grouped into three categories, depending on the role they play in the engineering decision-making process. The three categories are discussed in the following sections.

Diagnosticians. As stated earlier, most existing civil engineering KBES fall into the category of diagnosticians, that this, they tend to cluster around the derivation end of the problem-solving spectrum introduced in Section 2.

These KBES can provide advice in the form "What is wrong with the patient, or structure" (diagnosis) and can be naturally extended to provide "how to fix" recommendations (prognosis). The key characteristics of these systems are:

- the knowledge base contains *all* hypotheses or goals that the system "knows about" as well as the chain of inference or reasoning leading to each hypothesis, including all the symptoms or data needed to evaluate the hypotheses;
- for *each* of the possible diagnosis hypotheses, the knowledge base contains *all* relevant prescriptions and remedial measures appropriate for that hypothesis; and
- the inference strategies appropriate to the task are well known (e.g., use forward chaining if there are a few symptoms and many possible hypotheses; use backward chaining if the reverse holds; use mixed initiative if there are few key symptoms, chain forward to partial hypotheses, then chain backward to gather and evaluate confirming evidence).

The two emphasized "all" above refer to the current contents of the knowledge base; obviously, knowledge acquisition facilities are needed to expand or modify the knowledge base over the life of the expert system to reflect new or changed conditions or additional hypotheses.

There are three reasons for this predominance. First, many of the available expert system development *shells* are a direct or indirect outgrowth of diagnostic expert systems such as *MYCIN* or *Prospector,* and are therefore geared to supporting diagnostic problem-solving strategies. Second, this problem-solving strategy does correspond to a well-established mode of thinking or *paradigm* identified with the terms "engineering method" or "scientific method", namely, that a thorough analysis or diagnosis of the problem at hand facilitates the subsequent synthesis or prescription of a solution. Third, the paradigm provides for a natural growth of the knowledge base through experimentation and calibration: the knowledge base can grow in depth as the chains of inferences leading to particular hypotheses are elaborated, or in breadth as additional hypotheses and remedial measures are incorporated.

Diagnosticians can be either stand-alone KBES, or they can be interfaced with algorithmic programs or databases, as discussed previously. In particular, they can serve as intelligent pre- and post-processors for complex algorithmic programs. It is notable that two of the earliest KBES using general shells were civil engineering applications for modeling or pre-processing assistance.

HYDRO [Gasching81] was a KBES developed using the *Prospector framework* to aid hydrologists in generating numerical parameter values required as input to an algorithmic watershed hydrology simulation program. The knowledge base of *HYDRO* consists of a large number of small inference networks that represent the expert's knowledge for generating the individual input parameter values.

An early expert system addressing some aspects of an automated consultant which advises nonexpert engineers in the use of an algorithmic finite element program program was *SACON* [Bennett 78]. The structural mechanics knowledge base of *SACON* consists of: (1) rules for inferring analysis strategies, consisting of the identification of the most appropriate analysis class to be performed and associated analysis recommendations; (2) rules for inferring the controlling stress, deflection, and nonlinear behavior of substructures; and (3) mathematical models, in the form of procedural attachments, for estimating non-dimensional stress and deflection bounds for each substructure, based on its boundary conditions and loading. *SACON* was never intended for production use. Its sole purpose was to evaluate the *EMYCIN* environment for diagnostic applications other than the original domain of the medical diagnostic system *MYCIN*

Generators. The second category of KBES of great interest is that of generators, corresponding to the formation end of the spectrum. In contrast to diagnostic KBES, generative KBES can provide advice in the form "What is a feasible (or good or best) arrangement of entities subject to a set of constraints'*. The entities may be actions, as in planning, or physical objects, as in design. The constraints may be problem dependent, and typically propagate dynamically (e.g., choosing a particular action at one point will constrain the choice of actions at some other point). The key characteristics of such systems are:

- the constraints are represented in the knowledge base;
- the set of known entities, or rules for generating them, are in the knowledge base; therefore, these systems will not "invent" new plans or designs, but,may still generate novel candidate solutions by arranging or combining the known entities in unexpected ways.
- while the choice of inference strategy is not as clear as for diagnostic KBES, a number of inference strategies have been proposed or evaluated (e.g., hierarchical decomposition, least-commitment principle, means-ends analysis, etc) and can serve as a prototypes.

The "best" candidate solution is understood to mean best among candidates that can be generated from the known entities, measured according to some evaluation function. Again, the knowledge base can grow by the addition of new candidate entities, new constraints or new rules for combining entities.

There are understandable reasons why generative KBES have been slow to emerge: the early KBES development shells did not support this paradigm, and it takes considerable effort to decompose a design problem into a form amenable to this approach. The decomposition involves both decomposing the domain knowledge into a hierarchy of representations at various levels of abstraction, and decomposing the process or control knowledge into operations on these representations.

A prototype precursor of KBES in this category is *HIRISE [Maher* 84]. *HIRISE* synthesizes feasible structural systems from a known hierarchy of systems, subsystems, and components in a depth-first manner. Infeasible combinations are eliminated either by heuristic constraints (e.g., IF numbers of stories > 20 THEN eliminate rigid frame) or by the failure of feasibility constraints (e.g., IF uplift on windward column > reaction due to dead load THEN eliminate braced frame). Alternatives that are not eliminated are evaluated on a number of heuristic features (e.g., cost, speed of erection), and the alternative with the lowest evaluation function value is judged the "best". The output of *HIRISE* is a selected structural configuration complete with preliminary component parameters, suitable for further evaluation and detailed analysis and design.

A similar KBES, which includes the generation and solution of a finite element model, is *CARTER,* which designs engine transmission housings [Reynier 86]. It generates the shape based on technological constraints (e.g., clearances and rigidities needed by the gear train to be enclosed and supported), performs a preliminary sizing of the components, generates and solves a finite element model, and performs a limited amount of redesign on the basis of the computed response.

Critics. There is a great potential for a special class of diagnostic KBES, which may be called critics. The role of these KBES can be explained as follows. Generative KBES such as *HIRISE* and *CARTER* use preformulated constraints to guide the search for feasible solutions. However, there are many "soft" constraints that may affect the feasibility or optimality of a candidate design or plan which are not known with sufficient precision to be formulated as prior constraints. These soft constraints are not in the knowledge base of the designers, but are typically part of the expertise of the constructors, manufacturers or users of the system or product being designed. These experts, in turn, cannot be expected to tell the designer how to design something so that their constraints are satisfied; all they can be expected to do is to evaluate or *critique* a proposed candidate design and assert whether their constraints are satisfied or not (and, if not, then provide a reason or justification). Presumably the designer, when presented with such a critique, can modify his design so as to eliminate any cause of negative criticism.

One can conceive of many useful KBES that could perform the role of such critics. There is a lively interest in "design for manufacturability, constructability or operability" where *downstream* concerns of manufacturability, etc. are dealt with in the initial design. At the present state of knowledge about such concerns, an attractive implementation would be to pass candidate designs to KBES critics, and feed back the resulting criticism to the designers. For example, a constructability critic KBES could diagnose a proposed design and return statements such as "girder is too long to be lifted in place". It would then be the task of the designer (or design KBES) to modify the design so as to satisfy the implied constraint. Critic KBES would be particularly attractive in civil engineering, where design and construction are often performed by separate organizations, so that designers don't receive direct feedback concerning the constructability of their designs.

5. Evolution of KBES

The present generation of KBES has been justly criticized on two grounds: that they are *idiosyncratic* and that they are *static.* These terms require some explanation.

A KBES developed using the present methodology, described in Section 2, is *idiosyncratic* in the sense that its knowledge base represents the expertise of a single human domain expert or, at best, that of a small group of domain experts. The KBES thus reproduces only the heuristics, assumptions, and even style of problem solving of the expert or experts consulted. It is the nature of expertise and heuristics that another, equally competent expert in the domain may have different, or even conflicting, expertise. However, it is worth pointing out that a KBES is useful to an organization only if it reliably reproduces the expertise of that organization.

Present KBES are *static* in two senses:

- the KBES reasons on the basis of the current contents of its knowledge base, i.e., it does not "learn[1]* in the sense of automatically updating the knowledge base; a separate component, the *knowledge acquision facility* is used to add to or modify the knowledge base; and
- at the end of the consultation session with a KBES, the context is cleared, so that there is no provision for retaining the "memory" of the session (e.g., the assumptions and recommendations made).

These two characteristics of KBES result from the fact that the present KBES methodology is essentially based on AI research results of a decade ago. AI research has been steadily progressing during this time. The purpose of this, highly speculative, section is to attempt to predict the evolution of KBES as further research results of AI, as well as the experience in the development and use of KBES is incorporated into the next generation of KBES methodology.

At present, there appear to be no usable, formal methods for resolving the idiosyncratic nature of KBES. There are some techniques for checking the consistency of knowledge bases, but these techniques are largely syntactic (e.g., identifying rules with common premises but different actions). The growth of the KBES methodology from the present, highly "special purpose" expert systems to higher levels of generalization has to come from the profession or domain itself through research. Two possible avenues present themselves.

In the simpler, more passive approach, a researcher may make arrangements with the authors of several KBES in the same domain to obtain access to their respective knowledge bases and attempt to extract generalizations from them. This approach may be very frustrating because of the great variety of KBES development languages and their lack of representation of causal linkages between premises and actions.

A more active approach would start with the development of a domain-specific *meta-shell* which would contain: (1) the common knowledge base of the domain; (2) excellent knowledge acquisition facilities for expansion and "customization" of that knowledge base by a wide range of practitioners. Such an approach would have two advantages:

- individual organizations could develop their KBES more rapidly, by having as a starting point an initial common knowledge base; and
- evaluation of individual expertise and search for generalizations would be greatly facilitated by having a common representation for the domain knowledge.

A cooperative effort based on this approach has been proposed for expert finite element modeling and interpretation assistance ([Fenves 85], [Fenves 86]). A preliminary pilot system is currently being planned.

In contrast to the specialization vs. generalization issue, the issue of static vs. dynamic KBES appears riper for further improvement. There has been a great deal of research in AI on the topic *machine learning,* and some of the AI approaches are on the verge of being incorporated into engineering KBES. One of the most promising approaches is that of *leaning by analogy.* In this approach, the general objects or concepts in the knowledge base are described by heuristics features, and the KBES contains additional knowledge for classifying specific objects by their features. If there is a close match, i.e., if a specific object is analogous in some defined sense to a general object, then the rules pertaining to the general object are applicable. A contrasting AI technique, which may be termed *learning by generalization* deals more with the operations on the object. Initially, any major operation or *goal* of the KBES is typically decomposed into its constituent parts or *sub-goals.* Every KBES tries the various subgoals until it satisfies the original goal. A KBES incorporating learning by generalization does one more thing: it extracts from its context the set of conditions or premises that led to the successful solution of the original goal by the particular combination of subgoals. This new, learned combination of operators is stored as a "meta-rule"; if the set of premises presents itself again, the new meta-rule is directly invoked.

It is to be expected that these and other AI machine learning techniques will begin to be explored in the context of civil engineering KBES in the near future. The impact of this new generation of KBES will be most pronounced in two areas. One area is in recognizing precedents, where previous successful designs can be used as precedents for initializing new design activities by exploiting analogies between them and thus focusing design largely to that of "debugging", i.e., reconciling differences between the current design problem and analogous previous ones. A second area is that of elevating the critics described in the previous section to serve as components of generators, that is, to extract from their passive criticisms constructive constraints that can be incorporated directly in the formative, synthesis stage.

6. Summary and Conclusions

The methodologies and concepts of AI, as embodied today in the methodology of KBES, provide an intellectual framework for addressing heuristic problems in civil engineering that could not be successfully approached with conventional programming techniques based on well-structured, algorithmic formulations.

The scope and capabilities of the current generation of KBES may prompt the observation that AI in general and KBES in particular have been "oversold[1]*. Many of the present civil engineering KBES address trivially small problems, and very few KBES are in production use. This situation is to be expected in the early, formative stages of a new methodology. Current KBES development frameworks or shells were not motivated by engineering needs, and the current state of engineering expertise is not compiled in a form immediately suitable for incorporation into the knowledge base of a KBES. The stage of KBES today parallels that of algorithmic computing 30 years ago, that is, before the emergence of languages such as *FORTRAN* and *ALGOL,* in terms of both primitive development languages and primitive available knowledge. The reader is reminded that even the most straightforward algorithms, such as those for sorting or equation solving, have improved by many orders of magnitude from the original ones.

The present generation of engineering KBES have already had a major impact. First, they have clarified the distinctive needs of engineering KBES in four areas:

- reasoning with causal knowledge;
- interaction with algorithmic components;
- interaction with databases; and
- reasoning with spatial attributes and relations.

Second, the role of engineering KBES is becoming clearer. While the majority of present KBES are stand-alone systems addressing diagnosis or interpretation, KBES applications will broaden into:

- intelligent pre- and post-processors;
- critics providing feedback on proposed designs; and
- generators of designs or plans.

There is the further opportunity of progressively incorporating additional research results from AI research, particularly in the area of machine learning.

The third and most important lesson learned from experimentation with the present generation of KBES is the revision of the "classical" concept of knowledge engineering. Today's KBES development methodology is predicated on the presence of a knowledge engineer serving as an interface between the domain expert and the KBES development environment. This is again analogous to programming before *FORTRAN,* when an experienced machine language programmer was needed as an intermediary. It is not difficult to predict that history will repeat itself, and that the need for such an intermediary will largely disappear. It is a foregone conclusion that with a new generation of KBES development environments, application programmers in a domain will not only become the "knowledge engineers", but will be able to incorporate significant components of the domain knowledge base by themselves, relying on true experts only for key ideas and high-level heuristics. In this fashion, KBES will become another integral component of computer-aided engineering significantly extending computer-aided engineering, from pure calculating to true reasoning.

References

[Amarel 78] Amarel, S., "Basic Themes and Problems in Current AI Research," *Proceedings, Fourth Annual AIM Workshop,* Ceilsielske, V.B., Ed., Rutgers University, pp. 28-46, June, 1978.

[Baker 87] Baker, N. and Fenves, S.J., "Spatial Representation Language for Structural Design," *Expert Systems in Computer-Aided Design,* IFIP WG 5.2 Working Conference - Submitted to Conference, Sidney, Australia, February, 1987.

[BalzerSO] 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,* Palo Alto, CA, pp. 108-110,1980.

[Bennett 52] Bennett, J.M., *Structural Calculations on the EDSAC,* unpublished Ph.D. Dissertation, Cambridge University, Cambridge, England, 1952.

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

[Davis 81] Davis, R., et al., "The Dipmeter Advisor: Interpretation of Geologic Signals," *Proceedings, Seventh International Joint Conference on Artificial Intelligence,* Vancouver, B.C., pp. 846-849, August, 1981.

[Duda 79] Duda, R.O., et al., *A Computer-Based Consultant for Mineral Exploration,* Final Report SRI Project 6415, SRI International, 1979.

[Dym84] Dym, C.L., *Expert Systems: New Approaches to Computer-Aided Engineering,* Technical Report ISL-84-2, Xerox Palo Alto Research Center (PARC), Palo Alto, CA, April 1984.

[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,* Association for Computing Machinery (ACM), Vol.12, No. 2, pp. 213-253, February 1980.

[Fagan 79] Fagan, L.M., et al., "Representation of Dynamic Clinical Knowledge: Measurement Interpretation In the Intensive Care Unit," *Proceedings, Sixth International Joint Conference on Artificial Intelligence,* Tokyo, pp. 260-262, 1979.

[Fenves 85] Fenves, S.J., "A Framework For A Knowledge-Based Finite Element Analysis Analysis, Applications of Knowledge-Based Systems to Engineering Analysis and Design," *American Society of Mechanical Engineering, Special Publication AD-10,* 1985.

[Fenves 86] Fenves, S.J., "A Framework For Cooperative Development of a Finite Modeling Assistant," *Reliability of Methods for Engineering Analysis,* Bathe, K.J., and Owen, D.R.J., Eds., Swansea, U.K., Pineridge Press, pp. 475-486,1986.

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

[Forgy 84] Forgy, C.L., *The OPS83 Report,* Technical Report CS-84-133, Computer Science Department - Carnegie Mellon University, Pittsburgh, May 1984.

[Garrett86] Garrett, Jr., J.H., and Fenves, S.J., *A Knowledge-Based Standards Processor for Structural Component Design,* Technical Report R-85-157, Department of Civil Engineering, Carnegie-Mellon University, Pittsburgh, PA, September 1986.

[Gasching 81] Gasching, J., R. Reboh and J. Reiter, *Development of a Knowledge-Based System for Water Resource Problems,* SRI Project 1619, SRI International, 1981.

[Hayes-Roth 83] Hayes-Roth, F, Waterman, D., and Lenat, D., Eds., *Building Expert Systems,* Addison-Wesley, Reading, MA, 1983.

[Howard 86] Howard, H.C. and Rehak, D.R., *Interfacing Databases and Knowledge-Based Systems for Structural Engineering Applications,* Technical Report EDRC-12-06-86, Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA, October 1986.

[Kim 86] Kim, S.S., Maher, M.L. et al, *Survey of the State-of-the-Art Expert/Knowledge Based Systems in Civil Engineering,* Technical Report P-87/01, U.S. Army Corps of Engineers Construction Engineering Research Laboratory, Champaign, IL, October 1986.

[Livesley 66] Livesley, R.K., "The Pattern of Structural Computing: 1946-1966," *International Symposium on the Use of Electronic Digital Computers in Structural Engineering,* University of Newcastle, England, 1966.

[Maher84] 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.

[Maher86] Maher, M.L., "Problem Solving Using Expert System Techniques," *Expert Systems in Civil Engineering,* ASCE, pp. 7-17, April, 1986.

[McDermott80] McDermott, J., *R1: A Rule-Based Configurer of Computer Systems,* Technical Report CMU-CS-80-119, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, 1980.

[Nau83] Nau, D.S., "Expert Computer Systems," *Computer,* IEEE Computer Society, Vol. 16, pp. 63-85, February 1983.

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

[Reboh81] Reboh, R., *Knowledge Engineering Techniques and Tools in the Prospector Environment,* Technical Report 8172, SRI International, June 1981.

[Reynier 86] Reynier, M., "Interaction Between Structural Analysis Know-how and Chain of Reasoning Used by the CARTIER Expert System for Dimensioning," *Reliability of Methods for Engineering Analysis,* Bathe, K.J., and Owen, D.R.J., Eds., Swansea, U.K., Pineridge Press, 1986.

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

[Simon 81] Simon, H.A., *The Sciences of the Artificial,* Second Edition, MIT Press, Cambridge, MA, 1981.

[Smith 83] Smith, R.G. and R.L. Young, "The Dipmeter Advisor System,"IJCAI, 1983.

[Stefik81] Stefik, M., "Planning with Constraints (MOLGEN 1)," *Artificial Intelligence,* Vol.16, pp. 111-140,1981.

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

[Zumsteg 85] Zumsteg, J.R. and D.L. Flaggs, "Knowledge-Based Analysis and Design for Aerospace Structures," *American Society of Mechanical Engineers Special Publication AD-10,* pp. 67-80,1985.

Table of Contents

1. Introduction	0
2. Artificial Intelligence and Knowledge-Based Expert Systems	2
3. Distinction of Engineering KBES	8
4. Role of Engineering KBES	11
5. Evolution Of KBES	14
6. Summary and Conclusions	17