Expert systems: expectations versus realities

Introductory Keynote Lecture

Expert Systems: Expectations versus Realities

Systèmes experts: espoirs et réalité

Expertensysteme: Erwartungen und Wirklichkeit

Steven J. FENVES
Prof. of Civil Engineering
Carnegie Mellon University
Pittsburgh PA, USA

Steven J. Fenves, received his degrees in civil engineering from the University of Illinois, where he taught until 1972. His teaching and research activities deal with computer aided engineering, with emphasis on representation of standards, databases and expert systems.

SUMMARY
Expert systems offer a methodology for the treatment of heuristics in engineering computer-aided tools. The opportunity of incorporating heuristics has created high expectations. In reality, current expert systems are limited by the difficulty of knowledge acquisition and their static nature. The paper reviews the trends in expert systems development and identifies some promising explorations towards their generalization.

RESUME
Les systèmes experts offrent une méthodologie pour utiliser des «heuristiques» dans des systèmes d'ingénierie assistés par ordinateur. La possibilité d'inclure ces heuristiques a créé de grands espoirs dans les milieux concernés. En réalité, les systèmes experts existants sont limités par des difficultés d'acquisition de connaissance et par leur propre nature statique. Cet article donne un compte rendu des tendances actuelles du développement des systèmes experts et énonce quelques aspects prometteurs relatifs à leur généralisation.

ZUSAMMENFASSUNG
Expertensysteme bieten eine Methodologie an für die Behandlung von Erfahrung bei der Aufbereitung von Computer unterstützen Werkzeugen. Die Möglichkeit Erfahrung zu verarbeiten hat hohe Erwartungen geweckt. In Wirklichkeit sind jedoch gegenwärtige Expertensysteme beschränkt infolge der Schwierigkeiten der Wissensaquirierung, und durch die Tatsache, dass sie statisch sind. Diese Arbeit überblickt Tendenzen in der Entwicklung von Expertensystemen und zeigt vielversprechende Forschungsarbeiten in Richtung von verallgemeinerten Systemen.

1 Introduction

Developers of computer-aided tools have struggled for a long time with mechanisms for incorporating *idiosyncrasies* in their programs: assumptions, shortcuts, simplifications based on *experience*, and not necessarily fully justified by *causal reasoning*. Conventional (i.e., procedural or algorithmic) programming tools have made it difficult, if not impossible, to make the idiosyncratic or *heuristic* components of programs (a) *transparent* to users and (b) conveniently modified when the heuristics are changed.

The emergence of expert systems, or *knowledge-based programming paradigms* in general, have offered the promise of a methodology for better treatment of heuristics in engineering computer-aided tools. These methodologies rely, in essence, on modular and *declarative* representation of knowledge and thereby provide a means of overcoming the difficulties mentioned above. Thus, expert systems have been enthusiastically embraced in many disciplines, and have created high expectations among program developers, users and managers.

The realities of expert systems have not yet lived up to the expectations, for several reasons. First, the process of extracting, compiling and organizing heuristic knowledge, referred to as *knowledge engineering*, is turning out to be harder than anticipated, and present *knowledge acquisition facilities* are still rudimentary. Second, mature knowledge-based paradigms, implemented in languages, tools and shells, are available only for *interpretative* or *diagnostic* problems; the concepts and tools available for the more challenging *generative* or *formative* problems of design and planning are not yet well developed. Third, engineering applications invariably require a mix of qualitative (heuristic, empirical or "shallow") reasoning and quantitative (algorithmic, causal, or "deep") analyses and evaluations; the linkage between these two modes of problem-solving is not yet *seamless* enough for convenient and practical use. Fourth, realistic engineering applications require significant interactions with databases and with geometric and spatial representations; here again the necessary linkages are not yet available. Fifth, mechanisms need to be developed to extend "single-purpose", highly *idiosyncratic* expert systems to a broader scope. Lastly, present knowledge-based systems, derived from AI research more than a decade old, are *static* in the sense that they do not modify their behavior based on their performance; techniques of *machine learning* are only now beginning to be applied to engineering problems.

The paper reviews the trends in expert systems for civil engineering applications and identifies a number of explorations aimed at bringing reality closer to the expectations.

2 Background

The earliest 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 intervening years, computer use has undergone at least three revolutions:

- High-level procedural languages such as *FORTRAN*, *ALGOL*, etc., vastly increased the level at which engineers who could communicate their problem to the computer;
- 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 further 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; augmented by extensive user interface management capabilities, they have further elevated the level of interaction.

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 *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 technology transfer 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 physical, geometric 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 supports symbolic operands and heuristic operations. Existing design programs, developed in an algorithmic framework, contain 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.

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.

3 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 "spinoff" from Computer Science.

As early as 1956, 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 issues such as intelligent systems, search methodologies, knowledge representation, vision, natural language processing and robotics.

AI research on the representation and processing of knowledge has produced within the last decade 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 have raised the *expectation* of providingexactly the type of conceptual framework that is needed for the design and planning applications in civil engineering

The purpose of this paper is to explore how this expectation can be converted into reality.

4 Anatomy of KBES

An early definition of KBES, by one of the co-authors of the highly successful and influential *Prospector* system, is:

"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 [Gasching 81]."

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, as discussed above; and their output has long ago been accepted as advice, not as "the answer" to the problem.

A more satisfactory definition, elaborating on the role of expertise, is:

"An *expert system* is one that:

- handles real-world, complex problems requiring an expert's interpretation
- solves these problems using a computer model of expert human reasoning, reaching the

same conclusions that the human expert would reach if faced with a comparable problem [Weiss 84]."

Even this definition does not provide an operational definition to distinguish KBES from conventional algorithmic programs which incorporate substantial amounts of heuristics about a particular application area, or *domain*. The distinction cannot be based on implementation languages, e.g., *FORTRAN* vs. *LISP* (after all, many KBES frameworks are now implemented in *C*).

In an earlier paper, the following four key features that clearly separate a KBES from a conventional alrorithmic program were identified [Fenves 86]:

1. *Separation of knowledge-base control.* There are *some* facilities for manipulating the knowledge-base *per se* (displaying, searching, modifying) separate from the control (inference engine) which operates on the knowledge-base.
2. *Transparency of dialog.* There is some form of an explanation facility to convey to the user the inference process actually used. Conventional "Help" facilities do not qualify, as these are separate from the actual execution.
3. *Transparency of knowledge representation.* The domain-dependent knowledge incorporated in the program code is readable and understandable to some degree. Full natural language translation, as in *EMYCIN* and its derivatives, is not necessary; on the other hand, comments do not qualify, as they are not incorporated in the code.
4. *Incremental growth capability.* The KBES can be used with a subset of its ultimate knowledge base, and its knowledge base incrementally extended over a period of use without major (or any) restructuring.

The first feature needs some elaboration, since some KBES formalisms (e.g., frame-based systems) have no generic inference engine. A better generalization is to state that in a KBES knowledge is represented *declaratively* rather than *procedurally*.

Various KBES frameworks have different inference procedures and different knowledge representation schemes, including: production systems [Forgy 81], semantic inference networks [Reboh 81], logic representations [Clocksin 81], and frame representations [Wright 83]. More complex blackboard systems, based on multiple experts operating at different levels of abstraction, have also become common [Balzer 80, Erman 80, Nii 82]. 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, 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.

- *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 objective of the inference engine is to arrive at a global conclusion or *goal.* The inference 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 or shell 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 user 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.

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. 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.

5 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 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*, where a knowledge-based module is paired with each functional module of the algorithm, with the knowledge-based modules providing advice, checks, shortcuts, selections, etc., for their corresponding functional module [Fan 89].

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". 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).

Geometric Reasoning. A third distinguishing characteristic of most engineering KBES, particularly those in civil engineering, is their extensive use of spatial attributes of objects (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 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.

6 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. 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 4. A diagnostic 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 a 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* [Gasching 81] and *SACON* [Bennett 78]).

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.

Critics. There is a great potential for a special class of diagnostic KBES, which may be called critics.

The role of a critic is to evaluate a concept (e.g., a design or a plan) with respect to knowledge not available to the agent (human or computer program) that generated the concept [Talukdar 89]. The function of these KBES can be explained as follows. Generative KBES such as *HIRISE* [Maher 84] and *CARTER* [Reynier 86] 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 *a priori* constraints. These 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 critiques 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.

7 Critique of Present 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, 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" 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 and their acceptance or rejection by the user).

These two characteristics of KBES result from the fact that the present KBES methodology is essentially based on AI research results of more than a decade ago. AI research has been steadily progressing during this time.

The purpose of the following two sections 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.

8 Towards "General-Purpose" KBES

At present, there appear to be no usable, formal methodology emerging out of AI Research for resolving the idiosyncratic nature of KBES. There are some techniques for checking the consistency of knowledge bases, but these 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.

We are currently working on two projects using this approach. The first is the development of a PC-based diagnostic KBES for evaluating the seismic resistance of existing buildings [Fenves 89]. Because of the wide variety of expertise in this area, the basic system will contain only a core of commonly agreed knowledge (analytical routines, rules extracted from published documents, and reference facts) and a knowledge acquisition facility. A number of user organizations have agreed to experiment with the system and customize it by incorporating their own expertise. Arrangements have been made for these organizations to return to us their custom knowledge bases for us to collate and attempt to structure the accumulated knowledge.

In a second project, we are developing a finite element modeling and interpretation environment to generate numerical analysis models from suitable assumptions, and evaluate the numeric results with respect to these assumptions [Turkiyyah 89]. Provisions are made to customize the system by including user-specified preference rules for selecting among the assumptions.

9 Towards Learning KBES

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 of *machine learning*, and some of the AI approaches are on the verge of being incorporated into engineering KBES.

In a recent paper, we have surveyed a number of AI machine learning techniques and identified their applicability to KBES [Reich 89]. The potentially applicable techniques include: learning from examples; learning by analogy; learning by discovery; learning by experimentation; and learning by causal analysis. The applicability of these techniques lies in improving knowledge acquisition by building a domain model, increasing efficiency by learning search control, and reducing the "brittleness" of KBES by providing

common-sense reasoning. We are currently experimenting with a number of these techniques in the context of learning KBES for bridge design.

It is to be expected that 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 previously 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.

10 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. 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, 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 tens of 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 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 serve as 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 the present-day emphasis on calculating towards 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 and Functional Representation Language for Structural Design," *Expert Systems in Computer-Aided Design*, J.S. Gero, Ed., IFIP WG 5.2 Working Conference - Submitted to Conference, North-Holland, Amsterdam, pp. 511-530, February, 1987.

[Balzer 80] 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., 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.

[Clocksin 81] Clocksin, W.F., and Mellish, C.S., *Programming in Prolog*, Springer-Verlag, New York, 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*, Association for Computing Machinery (ACM), Vol. 12, No. 2, pp. 213-253, February 1980.

[Fan 89] Fan, Y., *A Knowledge-Based Finite Element System*, unpublished Ph.D. Dissertation, Department of Civil Engineering, Carnegie Mellon University, 1989.

[Fenves 86] Fenves, S.J., "What is an Expert System," *Expert Systems in Civil Engineering, ASCE*, In Kostem and Maher, Eds., New York, NY, pp. 1-6, 1986.

[Fenves 89] Fenves, S.J. and E. Ibarra-Anaya, "A Knowledge-Based System for Evaluation of Seismic Resistance of Existing Buildings," *Computer Utilization in Structural Engineering, ASCE*, pp. 428-437, 1989.

[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.

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

[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.

[Maher 84] 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 WG5.2 Conference, Budapest, Hungary, International Federation for Information Processing (IFIP), September, 1984.

[Maher 86] Maher, M.L., "Problem Solving Using Expert System Techniques," *ASCE Symposium On Expert Systems*, Kostem, C., and Maher, M.L., Eds., American Society of Civil Engineers (ASCE), pp. 7-17, April, 1986.

[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.

[Reboh 81] Reboh, R., *Knowledge Engineering Techniques and Tools in the Prospector Environment*, Technical Report 8172, SRI International, June 1981.

[Reich 89] Reich, Y. and S. J. Fenves, "The Potential of Machine Learning Techniques for Expert Systems," *Proceedings, AI EDAM*, 1989.

[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.

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

[Talukdar 89] Talukdar, S. and S.J. Fenves, "Towards a Framework for Concurrent Design," *ASME Conference on Concurrent Design*, 1989.

[Turkiyyah 89] Turkiyyah, G.M. and S.J. Fenves, "Getting finite element programs to reason about their assumptions," *Computer Utilization in Structural Engineering, ASCE*, pp. 51-60, 1989.

[Weiss 84] Weiss, S.M., and Kulikowski, C.A., *A Practical Guide to Designing Expert Systems*, Rowman and Allanheld, Totowa, NJ, 1984.

[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.

Leere Seite
Blank page
Page vide