ICHMT 2nd International Forum on
EXPERT SYSTEMS AND COMPUTER
SIMULATION IN ENERGY ENGINEERING
University of Erlangen, 17-20 March, 1992

ENGINEERING PROBLEM SOLVING WITH KNOWLEDGE-BASED SYSTEMS

WM. J. GARLAND
Department of Engineering Physics

W. F. S. POEHLMAN
Department of Computer Science and Systems

McMaster Univ.
Hamilton, Ontario, CANADA
L8S 4M1

ABSTRACT

This paper advocates that the nature of engineering problem solving is distinct enough that special consideration should be given to blending the new technologies of expert systems to the more traditional engineering technologies. The nature of both technologies are reviewed and some noteworthy conclusions are drawn. Two distinct engineering tasks are investigated by delineating their knowledge base structures and their solution methodologies. It was found that advanced expert system inferencing techniques are less useful in advancing the state of the art in engineering problem solving than carefully crafting the knowledge base structures.

INTRODUCTION

Most expert systems employ inference engines that are based on some variation of forward or backward chaining on IF-THEN rules; the 'search' is the key. The assumption is that this is a reasonable facsimile of how an individual problem solves. On the other hand, emulation of group problem solving is attempted through the Blackboard model. The paradigm there is that of parallel processing by specialized experts. Information is shared DURING the parallel processing. But do these models apply to the process engineer? What is the nature of problems faced by a process engineer and what strategies are appropriate? Are IF-THEN rules the hub of the domain expertise? Is the 'search' a key element in the problem solving process? How can group problem solving done? Herein, these issues are addressed and some insights are noted.

To illustrate some of the above we will consider:

1. The Selection of Heat Exchangers as an example of individual problem solving (focused on searching);
2. A Central Sampling Advisor for Nuclear Power Stations as an example of a more complex situation that benefits from a group problem solving approach.

BACKGROUND

Knowledge representation in artificial intelligence is firmly based on the concept of an object. This closely mirrors the human approach to visualizing the world; we clump things and concepts so that our active memory can deal with them. Our active memory can only handle four to seven items at a time. Hence, clumping is essential. Having proposed an object, we need to name it, set its attributes or describe it, put it into context with everything else, show how it interacts with other objects and put limits on the object. These activities are termed: NAMING, DESCRIBING, ORGANIZING, RELATING, and CONSTRAINING, naturally enough [PAR88]. NAMING simply puts one or more labels on the object. DESCRIBING provides a data base for information about that object. The common syntax is Object-Attribute-Value (O-A-V), for example: (car-wheels-4) assigns a value of 4 to the attribute, wheels, of the object, car. ORGANIZING provides a genealogy, taxonomy or classification scheme for the objects, such as the family tree. It is generally a static structure during the problem solution phase. RELATING focuses on the dynamic interaction with other objects. The way in which objects interact during the solution of a problem will change as the solution proceeds. For instance, deciding which car to buy entails a change in the relation between you (one object) and the set of all objects called cars. The ORGANIZATION of the cars in a classification tree has remained static, however. CONSTRAINING limits objects, usually through the use of rules.

The objects we select for representation depend on the problem to be solved. For the case of heat exchanger selection, discussed later, this gives approximately 30 to 50 objects. Defining the objects is an essential part of the problem definition and provides the basic information about which solutions can be formulated. The next step is to provide a means of manipulating the objects in order to formulate a solution.

For simple systems, we can simply use IF-THEN rules. IF-THEN rules have been shown to be sufficient for human thought modelling and for modelling all mathematical systems [PAR88]. Thus, theoretically, all problems can be solved this way, However, efficiency may be poor and the knowledge base may not lend itself easily to representation by IF-THEN rules only.

One can envision general structures of nodes (objects) and links between these nodes. These structures are termed semantic nets and can be of arbitrary complexity. No completely general implementation is commercially available. However, the currently available object based expert system shells are quite flexible.

Any real knowledge representation involves both symbolic and numeric processing. Hybrid types can be achieved by embedding or by blackboard techniques for distributed processing and a means of marrying generically different approaches and utilizing otherwise incompatible paradigms or knowledge representations.

Before discussing how to best represent typical knowledge bases in engineering applications, methods of manipulating the knowledge base to achieve the stated goals are discussed. This is known as inferencing.

Although knowledge representation (objects, frames, data bases, rules, etc.) forms the basis of the expert system, any expert system (humans included) requires a strategy for problem solving within the knowledge domain defined by the knowledge representation. Problem solving amounts to little more than searching, augmented by procedural calculations in most cases. Hence, searching methodologies form a focal point for most

inference engines. It is usual (see [WOL87] for a typical description) to subdivide search methodology into control strategy (providing the overall control) and the search tactics (providing the implementation details as requested by the control strategy).

The primary control strategies commonly used are forward chaining and backward chaining. Sometimes, both strategies are used in hybrid engines. Forward chaining involves starting with the facts and searching forward, using all possible rules, to generate all conclusions that can be derived from these facts. Backward chaining involves starting with an hypothesis and working backwards through the rules to discover if the hypothesis can be substantiated. One strategy is neither better nor worse than the other in general. Backward chaining is best suited to those problems that have a smallish set of possible hypotheses.

Search tactics refer to the details of the search through the set of rules. Given a network of rules, say in an inverted tree type structure that is 4 layers deep (trunk, branches, twigs and leaves), possible search paths include the following. A breadth-first approach searches all branches first before any twigs or leaves, then searches all twigs, and lastly all leaves). The depth-first approach picks a branch, searches a twig, leaf by leaf, then searches the next twig, and so on until that branch is exhausted, moves to the next branch, etc. Heuristic searches use rules of thumb to look in likely spots first. Hybrids of these approaches are, of course, also possible.

With this background, we now look at two domains that fall within the mandate of engineering.

HEAT EXCHANGER SELECTION

The first example is the task of selection. The task of designing is to a large extent one of making choices from a set of alternatives. Choosing the correct heat exchanger (HX) for a given application is such a situation. The experts here are engineers and, hence, tend to use first principles in their thought processes. On the most fundamental level, the mass, energy and momentum equations can be written for each phase of each stream as a function of space and time, coupled with their respective correlations and equations of state. But this large equation set cannot usually be solved even if all the constitutive relations were known. This level of detail is reserved for fundamental R&D where experimental and theoretical work are compared. The results of such scrutiny are practical correlations that are embodied into design and analysis codes. These practical tools permit the design engineer to scope out a number of designs in some detail. But this is not sufficient for good design for at least two reasons:

1. detailed calculations of the heat transfer, fluid mechanics and mechanical design are usually quite time consuming;
2. there are many other aspects to consider that affect the design, such as chemistry, fouling, past experiences, cost, reliability, service, etc.

Clearly, the designer cannot adopt a strategy which is entirely based on first principles, that is, design the heat exchanger starting from a clean sheet of paper; the task would be overwhelming in the intellect, time and effort required. The designer must be more pragmatic.

Heat exchanger design is now in a mature state; many viable designs exist covering the spectrum of applications. Designs have evolved around the flat plate (efficient in heat transfer, cheap to build, but inherently weak structurally) and the tube (strong mechanically at the expense of heat transfer and cost). Given the large number of variables in the design (materials, fabrication techniques, geometric details) there are a vast number of possible designs for heat exchangers. Time has eliminated most of the possibilities, leaving a manageable number (about 50) major classes of heat exchanger types that have proved successful in practice. The designer finds that the workable strategy is to first select a few candidates from these classes and to then refine the design using past experience to guide the design process. This is the heuristic approach; rules of thumb are used to guide the search, to narrow down the choices. Hence, the expert invokes a meta-knowledge, that is, a knowledge about his knowledge. The momentum equation need not be solved in detail for the expert to make some comment about pressure drop and how that affects selection. The knowledge captured for this project, therefore, has little to do, directly, with first principles. The primary focus must be the heuristics and supporting numerics that the expert uses.

In parallel with the maturing of heat exchanger design, the approach used by the experts has also matured and stabilized in overall structure (although improvements in design and analysis techniques mean that the actual designs are far from static). This has greatly aided the knowledge acquisition. There is an overall control strategy for HX selection that has emerged in the literature [LIN82, LAR83]. As in turns out, the strategy is highly procedural (meaning that the path to the solution is known) and invariant (meaning that the strategy does not change for different case studies). **This finding is very important to the project development.** It means that the HX selection process can be coded using standard procedural languages (BASIC, FORTRAN, PASCAL, C, etc.). If the strategy had been mainly declarative (meaning that the rules were known, ie declared, but the solution path was not known in advance), then AI techniques would have had to be employed. The AI approach could still be used but it would represent unnecessary overhead and reduce the speed of finding the solution considerably. A commercial shell was used at the beginning of the project but it was quickly discarded as unwieldy and unnecessary. A prototype has been written directly in 'C' and is depicted in Figure 1.

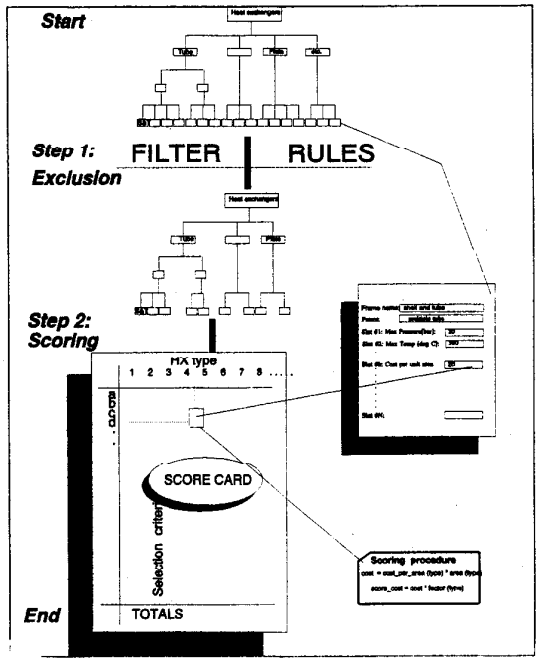

Figure 1 Heat Exchanger Selection Scheme

As discussed in [GAR90], the solution strategy centres around the elimination of HX's that cannot be used for one reason or another, and the ranking of those that are left by the use of score cards. Thus the first major task of the code will be to perform the exclusion. Exclusion rules could be applied sequentially to each HX in an exhaustive search fashion. A strategy is not needed for rule firing (inferencing) since an exhaustive search is being done. These rules utilize user supplied data, data from the HX data base, data from the fluid property data base, and derived data.

Process conditions form a major aspect used to select heat exchangers; consideration of the process conditions leads to most of the rules and constraints. For example, a typical statement from an expert is: if the pressure exceeds 20 bar then you cannot use a plate and frame HX because the gaskets cannot handle the pressure. It is convenient and efficient to collect all such values as 20 bar into a data base and loop through all HX's. This permits easy editing, enhances modularity, allows dynamic loading and unloading of data, and encourages generalization. The inference engine will use this generic rule to eliminate from consideration those HX's that cannot handle the pressures of the process streams of the application in question. In a similar fashion, generic rules have been formulated to exclude HX types based on: maximum heat duty; maximum pressure; maximum temperature; minimum temperature; number of streams; temperature cross; process attributes; stream attributes; fluid-HX incompatibilities; maximum heat transfer area; minimum heat transfer area; and special rules for HX's that require atmospheric pressure air or vapour streams or are subject to temperature approach restrictions.

After applying this set of exclusion rules, the inference engine is left with a considerably (hopefully) smaller set of possible HX's. The task at hand is now one of ranking this remaining set. The chosen method was score carding, in which each remaining object (HX) receives a score on a 100 point scale for a number of relevant attributes or constraints. These constraints or scoring considerations can be grouped conveniently into the categories of safety, process conditions, cost and other. Within each grouping are a number of sub-categories. Weights are assigned to simulate the relative weight to be given to each constraint. After the constraint scores are found, the weighted sum is calculated to give the total giving a ranking to the HX's that were not eliminated by the exclusion process. The score card approach provides an good way to perform the ranking since it emulates the expert methodology of weighing the pros and cons of the various HX's left on the list.

This approach will yield a set of individual rules, procedures, and constraints that are very transparent. It should be clear how each part functions since there is a very high degree of modularity and independence. The score for, say, the reliability of one HX can be set quite independently of all else and changes to that constraint gives a known and bounded change to the selection process. The whole process is very visible, having no iterative loops and employing a very high degree of separability of function in each of its main parts. Changing the exclusion rules will not effect the score card. Changing the scoring methodology cannot affect the exclusion rule functionality. Scoring details can be found in [GAR90].

Other schema, neural nets, rules with confidence factors and fuzzy logic, were contemplated early in the project. Neural nets were discarded at the beginning of the project because they are not good for telling WHY the conclusions were reached. Fuzzy logic seems inappropriate for the same reasons that confidence factors were ruled out early on: it is very difficult to adjust the likelihoods so that the rules combine correctly.

The prototype, constructed as an aid to knowledge acquisition and a precursor to the final product, effectively gives the topology of the knowledge base a tangible form. The user and the expert has a focus for discussion and the knowledge engineer has a tool for knowledge elucidation. The prototype performs very well, meeting or exceeding all expectations in ease of construction, ease of use, speed, and accuracy in emulating the human expert. The consensus among those interviewed was that the user should not be constrained by the program; rather, he or she should be free to move freely about the code segments. The user should control the code, not the other way around.

We next look at a quite different situation, one that involves real-time data acquisition, complexity, trend analysis, numerics and abstract decision making. We shall find that this requires a very different solution.

CENTRAL SAMPLING ADVISOR FOR NUCLEAR STATIONS

In an effort to aid the operator of large and complex plants, such as nuclear-electric generating stations and chemical process plants, an artificial intelligence based advisor is under development for the central sampling system of Pt. Lepreau Generating Station. Human expertise (inherently symbolic or pattern recognition based) must be married to the speed and pervasiveness of computerized data acquisition, trend analysis and simulation which all reside more in the domain of numeric processing. The schema advanced in this work achieves functional and temporal abstraction by using a tiered decomposition of the tasks, linked by real-time asynchronous agents acting through a blackboard. An overview is shown in Figure 2 [GAR89].

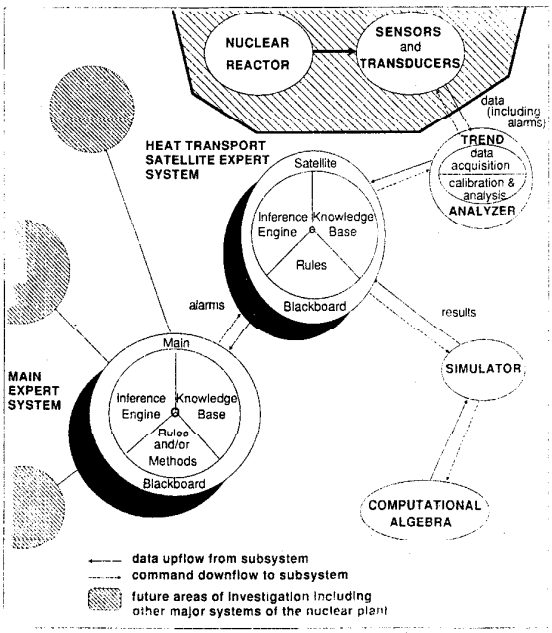

Figure 2 Operator Companion Overview

The objective of this research is to develop a specific instance in the small of functional and temporal abstraction for a real-time system using the anthropomorphic approach of a blackboard partitioned along the lines of manager - supervisor - technician:

Level 1: Manager (Planner) --> What needs to be done?
Level 2: Supervisor (Mediator)--> How is it to be done?
Level 3: Technician (Grunter)--> Do it!

This is not to say that the manager does not consider HOW or has nothing to DO, or that the other levels are equally singleminded. The levels merely state what the primary task of the various agents are. The manager does not need to know how a task is to be done or needs to monitor the details of the progress of the execution of the task. The manager only need to know that the task is doable and is started / in progress / finished, etc. The supervisor's primary task is to devise a plan of action and to supervise it's execution. The supervisor provides advise and feedback to the manager and technicians, thus the supervisor needs to be aware of Level 1 and 3 issues. Likewise, the technician is primarily responsible for the task execution but needs to be aware of the bigger picture to ensure the actions are consistent, provide feedback to the supervisor, etc.

One can use this schema for a given task as a modelling construct whether or not there are distinct agents for each level. For instance, for the typical office situation, there exists separate people for each level. But within each level, an individual (say a secretary), spends some of the time as a planner, as a mediator and as a grunter. The time or effort spend on each level would depend on the nominal role of the individual.

The Central Sampling Advisor, as currently envisioned, is composed of separate agents modelled along the lines of the above three levels.

The Manager decides what issue to look at. In this case:

Is there a leak? If so, how big?
Is there a chemistry problem? If so, what action is required?

The Manager poses the questions to the appropriate supervisors if they exist and are not busy (on a priority interrupt basis). If a specific supervisor is not available, then the questions can be posted to all supervisors in the hope that one can respond. The Manager monitors its incoming mail and reacts.

The Chemistry Supervisor responds to incoming requests (his inbasket) and reacts by posing two questions of its own:

Is the Condensate Polisher in service?
Does a transient condition exist in the plant?

The supervisor knows which technician can supply the answers to these questions. When the replies are received, the supervisor invokes one of four technicians to monitor a subset of the plant and chemistry data and to perform a fixed set of rules to determine the course of action. The Supervisor then posts messages to the Manager.

The Technician responds to incoming requests in a fixed manner. It needs to gather the required data and fire the rules of its domain in a forward chaining manner. The only difficult aspect is the time-varying nature of the data. Discussion with plant personnel revealed that, in general, it is not necessary to retain past history. It is sufficient to look at the current data and apply the domain rules. At the time of rule firing, flags are set to denote significant events. Thus, for leak detection, it is sufficient to record that a leak was detected and at what time it was detected. Subsequent analysis at a later time will thus have the needed knowledge of the event.

It is clear that the above activity of the various agents requires a central message facility. This is the domain of the Blackboard. Nominally, it consists of a message space (shared memory or files) controlled by a postmaster, whose duty it is to coordinate the flow of information amongst the various agents communicating with the blackboard. The blackboard contains the current knowledge state (status of questions posed, etc.). The postmaster locks and unlocks message space to prevent a READ during a WRITE and vice versa. While the postmaster is sequential in that it cycles through its 'route' repeatedly, the overall operation is asynchronous in that the sequencing of work is based on the messages passed, with no implicit requirement (or guarantee) that responses will be received at any given time or in any given order. Each agent proceeds independently to read its own mailbox, process information and send mail to the postmaster for delivery.

There is no forced and direct overall control of the duties to be performed. The Planner requests actions on a priority basis, based on its incoming mail, raising the priority as the immediacy increases. The Supervisor reacts to mail as it arrives and requests response from Technicians. The Technicians also react to their mail and request information from data acquisition agents. Messages are passed up the chain as results are achieved.

This morphology results in a natural decomposition of the functions to be performed (functional abstraction). Higher levels perform the more abstract operations. To provide the desired emulation of human experts, the Planner must be flexible. The algorithms are declarative in nature. These operations require more time to perform in general since searches through the domain have to be performed using backward chaining (goal driven) strategies. The antecedent - consequence matching required is time consuming and open-ended. Backtracking further compounds the time required to perform the higher level tasks. Lower levels are more non-declarative in nature. The time required to achieve a given task becomes more defined and the number of calculations per second increases as the agent spends more and more of its time doing, rather than thinking about what to do.

Thus, functional abstraction is not just a by-product of the schema; it is a REQUIREMENT if planning, mediation and execution is to be done in a real-time environment. The decomposition of the problem to levels and asynchronous agents gives the required decoupling of what is essentially a stiff system (to borrow an analogy from applied mathematics).

The morphology is also temporally abstract in that information hiding occurs. The higher level need not concern itself with information details of a lower level. The higher levels have less to do and more time to do them in. Again, this is not a by-product; it is a REQUIREMENT if planning, mediation and execution is to be done in a real-time environment. Higher level inferencing is, by nature, slower than lower level calculation. Information MUST be hidden for reasons of expediency.

DISCUSSION

It has been observed that experts are experts not so much because of their ability to perform mental gymnastics (inferencing) but because of their extensive knowledge base which includes much experiential history. Take experts outside their fields of expertise and their performance is not markedly different than non-experts. Human experts are experts apparently because of their ability to use their past experiences and knowledge as heuristics to quickly prune the search tree for the problem at hand. Thus the human expert focuses quickly on likely solutions

and doesn't spend time and effort doing fruitless tasks. He is usually not substantially faster at doing a specified task but he does only the necessary tasks. Compare this to the novice who must try many possibilities before finding plausible paths. This feature is exemplified by the HX example. It appears also in the Central Sampling Advisor example. There the skilled operator does not rely on a Sherlock Holmesian deductive capability to diagnose a problem. Rather, the plant and the plant controls, by design, has been functionally decomposed, as discussed, into systems and subsystems that are relatively simple and independent. This allows for the design of process control and safety systems that can be operated by trained personnel even in times of information overload and stress. Procedures and the use of heuristics are the order of the day. So, once again, the structure of the knowledge base and the knowledge contained within that structure form the backbone of the expertise.

Thus, determining the structure of the domain knowledge is a primary task. Our inquiry into this structure begins with the question: "What problem is to be solved?". Asking: "Is there a problem?" leads to an infinite search space. Asking: "Does problem X exists?" (where X is any of a finite set of known problems) gives a finite search space. If the space is small enough (and most practical engineering problems are) then exhaustive searches can be conducted, thereby eliminating the need for a search strategy altogether. Hence it is important that the question be properly posed. Thus we ask: "Which HX type is the best out of the 50 or so possible HX type?", not "Which HX type should be used?". And we ask: "Does a leak exist?", not "What plant fault exists?".

Next we ask: "How can the problem be solved?". Solutions are impossibly complex if you try to do a first principle design from scratch. The task becomes relatively simple if one utilizes the evolution of engineered designs to define objects to select from. The objects are selected based on an organized and accepted scheme, emulating the human expert's scheme. This allows a more meaningful emulation of rules and the user can more easily relate to the results. Genealogy or classification was NOT an issue in the actual inferencing, but it is important in that it aids the enumeration of the basic objects to be selected (possible outcomes). Thus, we evaluate the various attributes of the HX and produce a score as a measure of its goodness for the application at hand. And we invoke known operational procedures to determine the size of a leak or its location.

For both examples discussed herein, the what and the how questions were used to arrive at solution strategies. In fact, both problems have been 'engineered', giving double meaning to the title of this paper: 'Engineering Problem Solving...'. We solve engineering problems by engineering the solutions.

Continuing on with HOW, we note that the model is not the same as the human. Nor does it have to be. The achievement of an acceptable solution is the primary concern. But fidelity in emulation of the process of achieving that solution may be important for verification and validation purposes and for reassuring the user that the tool is a good one. For the Central Sampling Advisor, fidelity in emulation is important because the operator must be able to see the reasonableness of the recommendations given to him. The advisor is functionally decomposed along the same lines of reasoning that the operator has been trained to use. The HX case also attempts to follow the expert reasoning patterns for the same reason. But computers are not the same as humans. There is no reason to suspect that the same models would work with equal expediency in the human mind and in the computer CPU. The strengths of the computer model (fast in numerics, doesn't get tired or forget, and can be exhaustive) can be used to compliment the human expert (who is intuitive, has a dynamic and powerful heuristic, has a broader knowledge base, etc.). For the HX case, exhaustive searches replace heuristic searches. And for the Central Sampling Advisor case, vigilant data acquisition and trend analysis replace the operator's ability to see patterns in displayed data trends.

Emulation is also important for explaining WHY (or equally important, WHY NOT) a conclusion has been reached in terms that the user can relate to. The score card approach is very good in these respects because the user can directly see what factors are contributing to the conclusion. In general, the Decision Support System approach continues to be a good methodology to handle uncertainty and compliments the IF-THEN rule approach. It tends to be naturally WHY NOT. Unfortunately, fuzzy is fuzzy, no matter how it is packaged. No indisputable scoring procedures are available.

IF-THEN rules proved ideal for exclusion (collisions). The IF-THEN rule approach encounters problems with confidence factors and the score card approach proved superior as noted.

The modelling of the Central Sampling Advisor along anthropomorphic lines in an asynchronous manner permits parallel problem solving and the sharing of partial results during the solution process. On test cases, the degradation in performance due to the overhead involved in context switching and blackboard management is negligible. The three tiered layering, functional decomposition and temporal abstraction operating with the asynchronous blackboard should provide a means of marrying real-time data acquisition, numerics and symbolic reasoning. However, the concept has not yet been proven.

We have found that shells are useful as learning and scoping tools but ultimately are discarded. Invariably, the knowledge engineer starts out with a shell for initial discovery of the shape of the knowledge domain. Thereafter, coding takes place directly in a language. Embedded shells are a help but sooner or later, the bounds of a closed shell become too restrictive. This is especially true in engineering domains which are largely procedural, numerically intensive and are not symbolically intensive.

The coupling of symbolics and numerics is necessary and possible. The symbolic part need have little or no knowledge of the nature of the numerics. Therefore, this is a form of shallow coupling.

The paradigm must follow the user; otherwise the user won't be able to identify with the product or use it effectively. For the HX project, it was found that engineers tend to prefer that most input occur up front; Q&A is not appropriate since engineers usually work from a specification sheet of some sort. Q&A input is fragmented and disjointed. For the Central Sampling Advisor project, the plant operating procedures and control room displays largely define the format of the user interface and help bound the knowledge base.

The knowledge engineer can only get so far in simple verbal interviews. An early prototype is needed to enable efficient knowledge acquisition.

CONCLUSIONS

We have found that the expert has already a search strategy and this can be hard coded. Thus, the search algorithm to use herein is not a 'declarative' one, typified by forward or backward chaining. Rather, a 'procedural' or defined strategy (such as exclude and rank for HX and piecewise refinement for the advisor) is employed. As the prototypes mature, there may well be a future requirement for AI type declarative search strategies. However, the indications are that the bulk of engineering problem solving can be achieved in a procedural manner.

A typical problem involves a mix of procedures for searching and supporting calculations for the search. For engineering problem solving, much of the solution is numeric. This means that a collection of IF-THEN rules, no matter how large, is woefully inadequate for the 'number crunching' required. But equally, copious numeric output, no matter how large or correct, does not constitute a solution. The engineer reasons about the information contained in the calculations. Clearly, there is a need to couple the symbolics with the numerics. Engineering strategies are such that searching does not play a dominant role and, thus, optimizing the search strategy will result in little payoff.

Engineers engineer systems in a piecewise refined manner where simplicity is valued. Complexity is avoided. Declarative procedures are invoked whenever possible. In design, ie constraining, time has eliminated all but the successful evolutions - defining the pre-enumerated set of alternatives to choose from. In operations, operator overload, response time, etc., necessitate procedural responses. Successful operating plants are, by definition, procedural.

In short, the problem solving strategy usually appropriate to engineering problems is to select from among many alternatives (symbolic in nature), supported by calculations (numerics). This holds true whether the problem solving agent is operating as an individual or in a group. We have found that IF-THEN rules and search strategies do not play a major role in typical engineering problem solving.

So, is there no role for AI in the engineering domain? Yes and no. It depends how you define AI. One definition [BYL91] is: AI is the study of the relationship between computation and intelligence. If that definition is operative, then this whole paper is about AI and clearly there is a role for AI to play. But if one were to limit the definition of AI to the enterprise of designing computer systems that exhibit the characteristics associated with intelligence (understanding, learning, reasoning, solving problems, etc.), then there is not a major role for AI to play at present. Currently, the best use of computers in the engineering field is to offload the human by taking care of the mundane, repetitive or procedurally intensive tasks. This is what the typical engineer today needs and wants. This fits well with the human desire to remain in control. Both the HX and advisor projects revolve around the selection from pre-enumerated sets. Neither relies to any great extent on "traditional" AI or ES techniques. Rather, it is more expedient to resolve the code along the same functional lines as the human expert engineer has used - functional decomposition and piecewise refinement. For the problems investigated, search heuristics is neither a necessary nor efficient paradigm to use. The power of AI and ES inferencing is the flexibility in setting agendas, adapting to the resetting of goals, and ease of setting up and maintaining inference intensive solutions. Engineers tend to use static strategies and so the domain of engineering problem solving cannot easily capitalize on traditional AI and ES developments.

However, looking ahead, perhaps more complex invocations might lead to a different balance. Perhaps other engineering problems entail a significant element of synthesis (herein, there is none). Perhaps, there exists a more general paradigm, say a kernel of production rules that guide a 'little engine that could', that generate solution strategies on the fly, that solved engineering problems in a more generic manner. That would be AI.

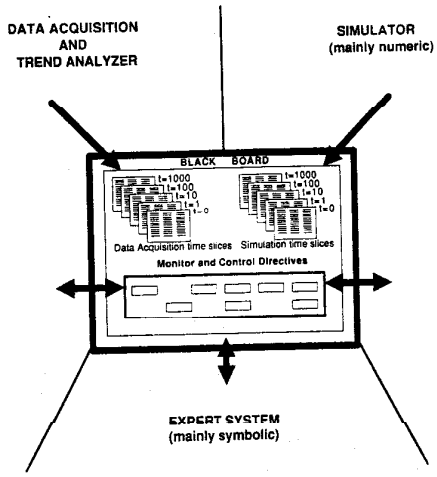

Figure 3 Blackboard Schema

ACKNOWLEDGEMENTS

This work has been made possible through funding from Atomic Energy of Canada Ltd. and the Natural Sciences and Engineering Research Council of Canada, with further facility and staff support from HTFS-Harwell, UK, HTFS-CRNL, Canada, and New Brunswick Electric Power Commission. The authors wish to express their sincere thanks to the many individuals who have freely given their time and effort in support of this project. Without exception, their willingness to provide input and their courteousness is unsurpassed.

REFERENCES

BYL91 T. Bylander, public electronic communication, 1991.

GAR89 W.J.Garland, W.F.S. Poehlman, N.Solntseff, J. Hoskins and L.Williams, "Intelligent Real-time Systems: Towards an Operator Companion for Nuclear Power Plants", Engineering Computations, 6, (1989) pp.97-115.

GAR90 Garland, Wm. J., "Knowledge Base Design for Heat Exchanger Selection", Engineering Applications of Artificial Intelligence, Volume 3, September 1990.

LAR83 Larowski, A. and Taylor, M.A., "Systematic Procedure for Selection of Heat Exchangers", Proc. Instn. Mech. Eng., Vol 197A, January 1983.

LIN82 Linnhoff, B., Townsend, D.W., Boland, D., Hewitt, G.F., Thomas, B.E.A., Guy, A.R., and Marsland, R.H., "User Guide on Process Integration for the Efficient Use of Energy", Institute of Chemical Engineering, 1982.

PAR88 Parsaye, K. and Chignell, M., "Expert Systems for Experts", John Wiley & Sons, Inc., ISBN 0-471-60175-6, QA76.76.E95P27, 1988.

WOL87 Wolfgram,D.D., Dear, T.J., Galbraith,C.S., "Expert Systems for the Technical Professional", John Wiley & Sons, 1987, ISBN 0-471-85645-2.