THE ROLE OF KNOWLEDGE-BASED SYSTEMS IN HEAT EXCHANGER SELECTION, DESIGN AND OPERATION

WM. J. GARLAND
Department of Engineering Physics
McMaster University
Hamilton, Ontario
CANADA L8S 4M1

Abstract

This paper explores the use of Knowledge Based Systems as tools to aid the user with engineering activities relating to heat exchangers. The areas of heat exchanger selection, design and operation are addressed. The nature and importance of knowledge base design is investigated. It was found that the knowledge base design is inextricably linked to the mental models of the user and of the expert being mimiced, and so, human problem solving strategies are investigated. Prototype designs for selection, design and operation have subsequently been built and have proved successful. The prototypes are discussed.

1. INTRODUCTION

The emergence of practical Expert Systems (ES) in recent years has led to their consideration for engineering activities. Expert Sysytems entail the combination of data-bases with rules for the manipulation of the data to form knowledge-bases. Engines are included to make inferences based on the knowledge. Knowledge-Based Systems (KBS) is perhaps a better term since it more accurately reflects the fact that most successful implimentations are empowered by the knowledge contained within the tool, rather than by novel engines that manipulate that Knowledge-Base (KB). Herein, we look at the nature of ES and KBS to delineate what they are good at doing and what they are not good at doing. Then, we look at the domain of heat exchangers with a view to assessing where these emerging tools might be of value for selection, design and operation.

Arguably, we can divide the problem domain up into 3 distinct parts:

1. Selection: given a job to do, which HX type should be used?
2. Design: given a HX type, what are the design specifics?
3. Operation: given an engineered HX, usually as part of a system, how should it be operated and maintained?

These demarcations are somewhat mutable since selection involves some design activities, design considers operational issues, and so on. There are other activities missed out altogether, such as process integration, project management, procurement and installation but they lie (mercifully) outside the scope of this paper. Specifically, we will look at prototype KB tools covering the HX selection, S&T design and turbine condensor chemistry support. But first, it is necessary to introduce KBS and user cognitive issues.

2. THE NATURE OF ES

Traditional numerically based computer codes for engineering are structured on the model:

Data Structures + Algorithms = Programs

The algorithms are invariant procedures for the most part, although some programs involve 50,000+ lines of code containing complex logic. Characteristically, the solution procedure is

intermingled with the domain knowledge within the code. In contrast, Knowledge-based systems (KBS) are structured on the model:

Knowledge + Inference = Knowledge-Based System

Knowledge is composed of data and the rules that apply to that data. Traditional programs would have the rules as part of the algorithm whereas modern KBS use an engine that is distinct from the knowledge-base to pursue the goal specified by the user. Knowledge, in essence, is the input data for the inference engine in the same manner as data structures are for traditional procedural codes. We illustrate the heirarchy of knowledge forms in Figure 1. In this sense, KBS operate one or two levels up from data based systems in terms of abstraction, making them one step closer towards emulating human expertise and one step farther away from dumb tools. By separating out the KB from the inferencing engine, it is now possible to build a very flexible tool; the knowledge, once captured, can be operated on in many ways. One need only declare the data and rules in the form of a KB; the solution procedure need not be supplied since the inference engine will provide the problem solving as needed. This approach is simply too attractive to ignore. Let us dig a little deeper to understand the advantages and disadvantages better.

2.1 Knowledge representation

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. The objects we select for representation depend on the problem to be solved. For instance, it could be the HX type, giving 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 [1]. Thus, theoretically, all problems can be solved this way. It is no wonder, then that symbolics became the backbone of AI research. However, efficiency may be poor and the knowledge base may not lend itself easily to representation by IF-THEN rules only. This is especially true when the rules are not definite, that is, when the situation is 'fuzzy'.

It has been observed and is widely recognized in the literature that it is the vast knowledge-base that exemplifies the expert rather than raw inferencing capability. Take an expert and place him or her in novel territory and you get decidedly non-expert behaviour. Take a genius and place him or her in the expert's territory and you do not get expert behaviour. The knowledge-base (composed of facts and heuristics) is then very important. The result of this is that the objects being manipulated (the knowledge base) need to be defined before they can be manipulated (by the inferencing or procedural engine). But how we define these objects will depend on the mental model chosen (a point that we will come back to later).

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 flexible enough to meet most needs today.

Methods of manipulating the knowledge base to achieve the stated goals are discussed next. This is inferencing.

2.2 Background on control and search strategies

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 [2] for a typical

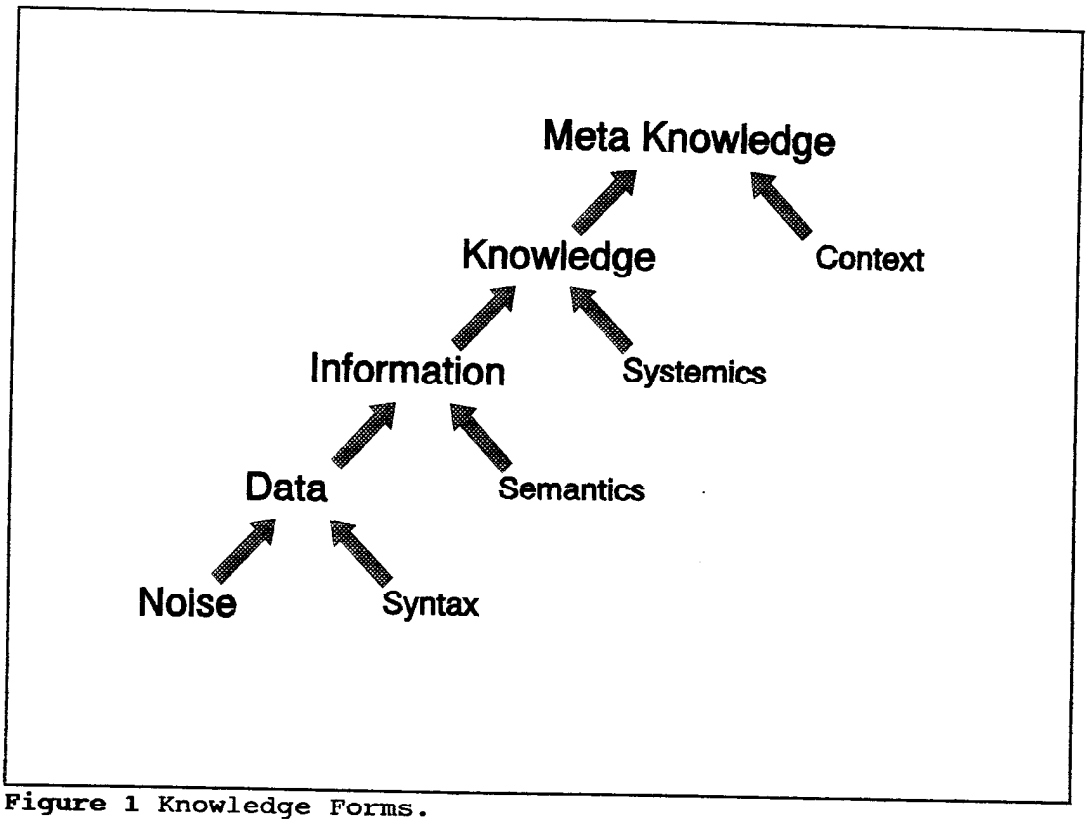


Figure 1 Knowledge Forms.

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, such as when the objective is to confirm a hypothesis or perform a diagnosis. On the other hand, forward chaining lends itself to data driven applications like real-time events. Both strategies can be augmented by first firing rules which prune the search tree, thus reducing the search space.

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 and to prune the search space. Hybrids of these approaches are, of course, also possible.

One can see, then, that given a set of IF-THEN rules that represent knowledge and given a goal plus initial facts, the solution can be found by invoking an inference engine that makes deductions without being given an explicit solution procedure for the problem at hand. This scheme is termed 'declarative' as opposed to the 'procedural' scheme traditionally used in engineering.

Any real knowledge representation involves both symbolic and numeric processing. Engineering

applications are typically numerically intensive. Hybrid types can be achieved by imbedding or by blackboard techniques for distributed processing and a means of marrying generically different approaches and utilizing otherwise incompatible paradigms or knowledge representations.

In short, KBS are distinguished from traditional programming by:

- The use of symbolics
- The ability to adapt to changing goals (declarative)
- the separation of the inference engine from the data (KB)
- the use of search techniques, including pruning

The lure of AI stems directly from these characteristics; we have the kernal of the elements needed for a thinking machine.

On the plus side, KBS are good for problems that are predominately symbolic in nature, have variable goals and have an unbounded solution space. In addition, because AI techniques are different from the traditional techniques, they involve generically disparate solution schemes and so can be used to verify and validate existing schemes. This has significant safety implications. AI based schemes are naturally adaptable to explaining why a conclusion has been reached. The many successful expert system shells incorporate explanation facilities, forward and backward chaining, and extensive development tools are standard repitoire. Hence, these tools provide excellent platforms for knowledge acquisition or domain exploration and rapid prototyping.

But all these benefits come at a price: speed and resources. Because the inference engine must formulate a game plan 'on the fly' and invariably follow many dead ends before a conclusion is reached, performance is slow when compared to the procedural paradigm. Further, these tools are not optimized to execute numeric calculations. There is a huge memory and disk space overhead for the inference engine and the associated environment. Consequently, such systems are not suited to real-time applications and are not well suited for engineering applications as we shall see. It is obvious, but often overlooked, that some mental activities like instinct cannot be captured by rules. Hence we cannot expect to use KBS for activities that involve instinct. If we cannot formulate our thoughts, we can hardly expect the machine to be able to emulate us. AI has been oversold, or more accurately, overbought. Vendors have emphasized the positives, true. But buyers tend to overlook the negatives, wanting to believe that here lies the answer to their problems. The hard reality is that, as before, the accuracy of the answer depends on the input data; all relevant knowledge must be represented in the KB. It turns out that elucitating the KB is the primary task; implimenting it in a system is usually straightforward.

As we shall see, even though there are limitations with KBS, they have their place and can be quite effective when used appropriately as compliments to traditional engineering tools.

Further distinction can be made in solution paradigms: individual problem solving vs. group problem solving. It has already been pointed out that the search is the primary strategy for problem solving and thus provides the focal point for inference engines. This applies to the individual. However, this does not emulate the primary mode of problem solving on a group level. We note that real problems solved on a group level are complex and diverse. The knowledge-bases are necessarily system or component specific and it is unlikely that a general knowledge-base can ever be conceived. This leads to the paradigm of message passing via a blackboard or some other mail handling system as a means of allowing disparate agents (models or codes or humans, etc.) to interact. Knowledge-base design is, in this case, more about the design of the blackboard - the format and content of the messages being processed. It is at this level that one is concerned about how the operator or engineer interacts with the system being controlled. The user's mental model is thus important for group problem solving. Recall, too, that since the mental model defines the opjects in the KB, the user's mental model is important for individual problem solving as well.

3. THE USER

Past work on Expert System based aids has been machine-centred. This is a term used by Bernard [3] to describe the overall design philosophy of building tools that put the machine in control, that

is, the computer program tells the user what to do and when to do it. Natural scepticism coupled with the clearly limited expertise exhibited by Expert Systems ensured failure for attempts at building machine-centred aids. Contrast this to the human-centred approach wherein the user is the primary source of intelligence and is in control. In the human-centred approach, the user employs the computer programs as tools, as powerful extensions of the user. The computer may monitor and annunciate but it is the user who pilots the operation, using the tools when necessary and as necessary. The paradigm shift is profound. Bernard notes that the machine-centred approach is now considered inappropriate.

The mental model of the designer or engineer as developed in Rasmussen's book [4] is one that is based on functional decomposition. The engineer poses: How does the plant work? What is broken? What measurements must be taken? What is the functional decomposition of the plant? How do the parts interact? How can one simulate it? This 'mechanology' led to alarm based annunciation, control room displays and controls grouped by system (functional decomposition), sensoritus, and information glut without enhanced knowledge. The view of the plant taken was that of the design engineer - this is how and why it works - here are all the details, etc. Ergonomics was commonplace but limited to 'knobology'. Make the knob bigger, use a red light here, etc. This kind of thinking leads to products like the VCR - machines with attractive lines, buttons that 'feel' right, on-screen programming and 64 button remote controls - full featured functionality from the comfort of your armchair. And unusable for anyone except a technoid! Problem solving strategies here relies on a deeper than average understanding and use of specific knowledge. This is appropriate for design activities but not for selection and operation activities.

The non-expert selection user or the plant operator have different mental models than the design engineer. The operator's mental model of the plant is closer to that of a technician - the plant is a collection of many systems, most of which are treated with generic algorithms for fault diagnosis and treatment of event symptoms, irrespective of the system in question. The non-expert involved in a selection exercise also behaves like a technician, employing general strategies irrespective of the system in question (elimination of inappropriate alternatives and evaluation of the remaining alternatives using common and measurable attributes). Rasmussen's figure (reproduced in part in Figure 2) illustrates this generic algorithm. The generic algorithm for problem solving is to observe and identify the state of the situation, interpret, evaluate, plan actions and execute the actions. Rasmussen notes that shortcuts can be taken at any stage. In fact, most of what we do involves shortcuts to some degree. ALL problem solving is covered by this figure but the technician often employs strategies and tactics that do not rely heavily on a detailed knowledge of system and component behaviour. That is, short cuts to Rasmussen's full solution path are taken. This is a form of shallow reasoning and is good most of the time. This is not to say that the operator does not have a detailed knowledge of the systems and components. He or she indeed does. It is simply that the problem solving scheme is not as system dependent as for the engineer.

Recalling that the KB definition is fundamental to the whole exercise and that the KB is defined around the objects of the problem domain, and having just learned that different user types employ different mental models (hence different fundamental objects) we sometimes face a delemma: the physical problem domain needs to be functionally decomposed along the lines of the physical or engineer's mental model since that's how it works, whereas, this is sometimes an inappropriate model for the user. The issue is not a trivial one and, as such, deserves careful consideration in the design of support systems. This mental model mismatch plus a machine-centred bias are arguably the leading causes of the failure of artificial intelligence based support systems.

4. HEAT EXCHANGER SELECTION

The first example of the use of KBS for HXs is the task of selection. The task of selecting is to a large extent one of making choices from a set of alternatives. Design considerations do play a significant part in selection since frequent forays into the design details are required even for the task of eliminating alternatives that are not feasible for the task at hand. Choosing the correct heat exchanger (HX) for a given application is such a situation. The typical user is an engineer

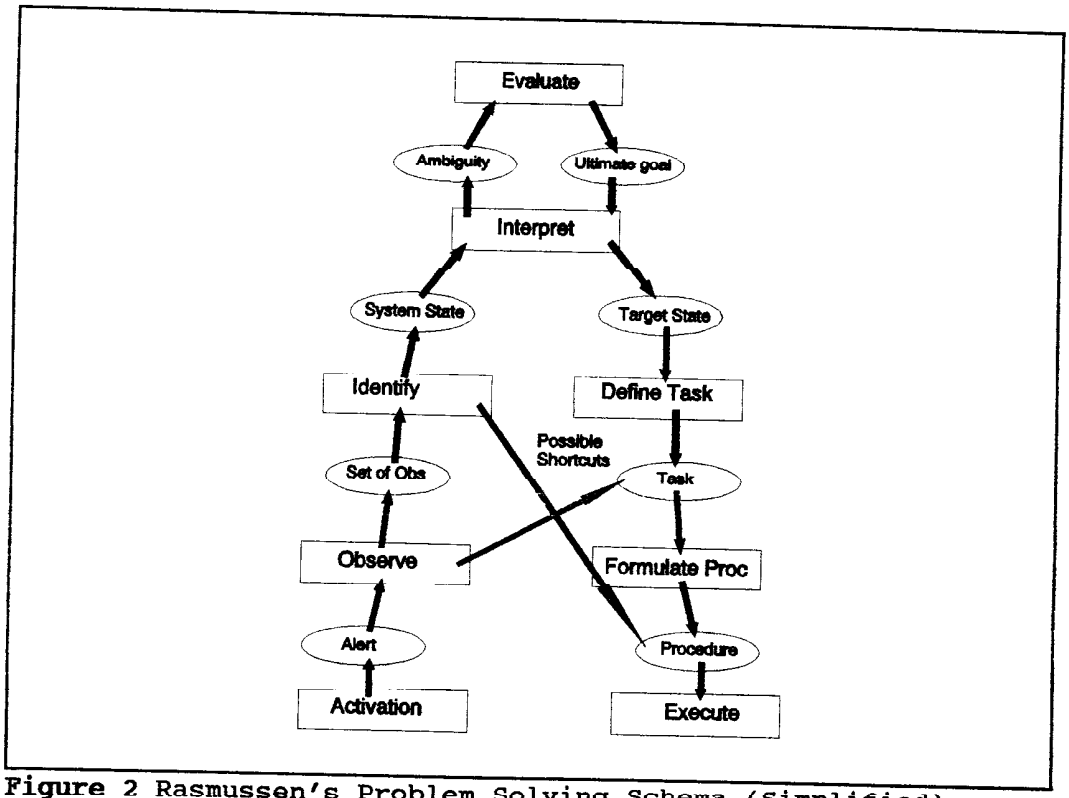


Figure 2 Rasmussen's Problem Solving Schema (Simplified).

who is not an expert in the HX field. The experts here are design engineers whose knowledge we hope to have captured in the HX selection tool. These experts tend to use first principles in their thought processes. To see how we might best organize our KB, let us look first at how a design engineer might approach the task of selection.

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 expert 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 expert 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 expert 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 selection, therefore, has little to do, directly, with first principles. The primary focus must be the heuristics and supporting numerics.

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 [5-7]. 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'.

As discussed in [8], 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 mainly on the basis of cost. 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 is currently cost based but development continues.

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 have 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.

In short, the task of HX selection involves selecting from a finite and pre-enumerated list of alternatives (hence the search for a solution can be exhaustive), the goal is fixed and the solution path is known (ie fixed invarient procedures), the solution path is straightforward, the solution

depends mostly on the knowledge base that is predominantly numeric, and the appropriate mental model is more akin to the operator than it is to the engineer. Compare this problem to the KBS based on the ES paradigm and we see that the match is not good at all. Traditional programming techniques can be used and an inference engine is not required. None of the features offered by KBS are needed except for the knowledge base itself. As mentioned previously, the user adopts the operator mental model for HX selection. Thus, the KB has been cast in terms of objects that the non-expert can associate with and the rules are easily understood heuristics supplied by the expert. The difficulty in building an HX selection tool lies in the knowledge acquisition task because of the disparity between the mental models of the user and the expert.

5. SHELL AND TUBE HEAT EXCHANGER PREDESIGN

The objective of the second exercise is to achieve a rough design for the S&T HX. But, as an objective, this is too vague. The design exercise is more specifically the selection of a number of design options that are needed as input into S&T design codes. The items to select, ie. the 'objects', are:

- stream allocation (hot stream to the tube side or shell side?)
- front head (A, B, C, N or D type?)
- shell (E, F, G, H, J, K or X type?)
- rear head
  - fixed tubesheet (L, M or N type?)
  - U tube (U type?)
  - floating tubesheet (P, S, T or W type?)
- number of tube passes (odd or even?)
- triangular or square tube pitch?
- shell orientation (vertical or horizontal?)
- expansion bellows (yes or no?)

There are the obvious groupings, front head, shell, rear head, etc., and only one member from each group can be selected at any one time (ie., the front head is A or B or C or ...). Further, the choice of front head is not independent of the rear head, etc.

There are a number of subgoals. The number of possible configurations is quite large, approaching:

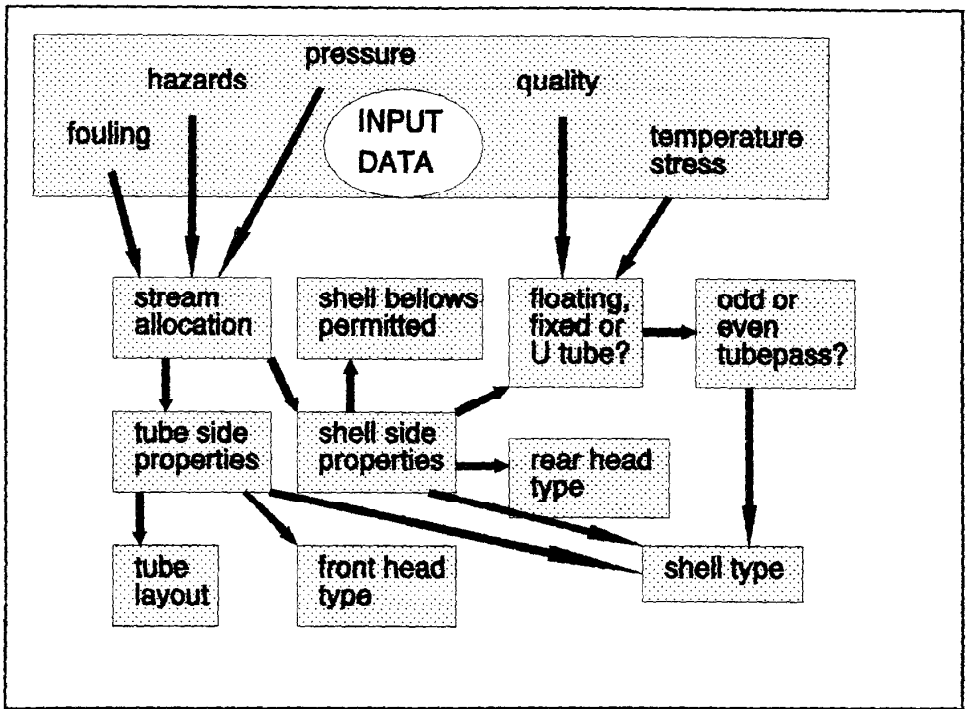
$$2 \times 5 \times 7 \times 4 \times 2 \times 2 \times 2 \times 2 = 4480.$$

Thus, an exhaustive search though all possible design configurations is not advised; nor is it the route followed by the human expert. The alternative approach is to apply heuristics (rules of thumb) to select the key design items, thus drastically narrowing down the search space.

An attempt was made to utilize the rules documented in [9]. Initially it was thought that AI techniques would be required since the logic seemed quite convoluted. However, with the use of an influence diagram (illustrated in part in figure 3), it became clear that, once again, a fixed procedure could be used. Consequently, a simple scorecard approach was used.

From the influence diagram, we can easily detect those items that could be selected based on the process input only, independent of the other design items. Stream allocation is one such item. In addition, all other design items depended on the stream allocation. Obviously, this should be the first item on the agenda. Once that is decided, the temperature stress situation guides the bellows selection. The ground work is now prepared for the choice of tubesheet (fixed, U tube or floating). The rest of the decisions follow readily. Thus the S&T design is modelled as a linear (non-iterative) application of independent modules: allocate streams, decide on bellows, etc. Since that the procedure has been clarified, there is no need for an inference engine. It is possible that future study will reveal that the design procedure is not as invariant as just depicted. For example, one application might require that stream allocation be left open until other aspects have been investigated. If that proves to be the case, then the firing of these independent modules can be retained, but controlled by a true declarative based inference engine that provides the overall strategy. There is no loss, then, in experimenting with the procedural approach.

These design items were found to depend on the following 7 process items to a greater or lesser



extent:

pressure
hazards
fouling
stream quality
stress due to temperature differences
cleaning requirements
interstream mixing

Other items, such as pressure drop and volumetric flow rate, also influence the decisions but these were not pursued at present.

As an example of some rules used in the pre-design task, consider the first task: allocation of the streams to tube side or shell side. Generally, high pressure streams should be placed on the tube side since shell side wall thicknesses would be prohibitively expensive if high pressure fluids were placed on the shell side. Hazardous fluids are likewise more easily accommodated on the tube side. Fluids that cause wall fouling (and thus heat transfer degradation) should be placed to facilitate mechanical cleaning, that is, on the tube side. Immediately, we see conflicting requirements: where should we place the streams if one is high pressure and the other is hazardous? The experts tell us that priority should be given to the high pressure rule, then the hazard rule and lastly the fouling rule. This is justified since there are alternative strategies for dealing with fouling fluids and the cost of high pressure shells far outweighs the cost of additional measures for dealing with hazardous fluids. We assign the rule weights to be 45%, 35% and 20%.

Next we must translate the stream pressures into some score (say on a scale of 0-100) representing the relative need to put that stream on the tube side. The same is done for hazards and fouling and a simple weighted sum is calculated for both the hot and cold streams. The stream with the highest score is allocated to the tube side. This process seems somewhat arbitrary and it is. However, a more elaborate procedure would not improve the accuracy because the limitation is with the expert knowledge itself. No hard and fast rules exist for stream allocation; the rules are

somewhat fuzzy.

The pre-design task as illustrated above is similar to the HX selection task; for the pre-design task, choices are made on design specifics, thereby narrowing down the number of possible design options and making the whole design process more efficient. In essence, one can think of the whole design exercise as a constraining exercise that takes the designer from a huge number of possibilities to one specific design. The whole design process is one that is virtually unbounded, has variable goals, requires intuition and is a very subtle process. This is a good match to the characteristics of the KBS except for the presence of intuition. However, as illustrated by the pre-design solution procedure, it is necessary to first capture the expertise in the form of a KB before an inference engine can be employed. But the very exercise of capturing the KB also exposes the solution procedure (that had seemed so ethereal before) to be, yet again, a simple procedure. Once more we see that an inference engine is not required; capturing the knowledge in the KB is the key.

The traditional practise is to supply good analysis tools. In the end, it is the design engineer who must remain in control of the design decisions make. What we can offer here is a tool to do a rough pre-design to aid in the use of the detailed tools. The mental model of the user is the engineer's mental model which makes the task of knowledge-base design that much easier.

6. OPERATION

The third example is in the area of operations. To respond to real events in a complex plant, analysis is required. Much of the control is automatic but a significant role is played by the operators who must reason about the situation at hand. The right hand side of Figure 4 illustrates the roles played by the human operator with respect to the plant and plant control (as portrayed in more detail by [10]). It is not just a question of being fast. Rather, bounding the solution time is the issue. Can a sufficiently good solution be found within the time required? The goal here is to 'sufficify' rather than 'optimize'. The successful operator or control system reaches the appropriate conclusion and implements it before the real system (the plant) does. Approximate solutions can be refined later. Here again, the heuristics are not well delineated.

In complex plants, the sub-systems of distributed architectures are typically hierarchically organized. Compared to higher level sub-systems, lower level sub-systems respond more quickly to more basic inputs in a more procedural manner. An example would be a standard proportional-integral-differential (PID) controller. Typically data flows upward through the hierarchy and at each level is transformed in some way. Higher level systems need not be informed about unnecessary information. This is information hiding. Higher level subsystems (the operator constitutes the highest level) involve more highly abstracted data which requires higher levels of cognitive analysis and which, by their very nature, are performed more slowly than lower level responses. This is temporal abstraction. Figure 5 illustrates the functional and temporal abstraction typically found in a complex plant.

The plant is organized along the same hierarchial lines as the design engineer's mental model. That organization expects the human to provide the top level knowledge based control. The operator, however, spends much of the time at the rule based and skill based levels. Figure 4 illustrates this point of mental model mismatch as previously discussed. Next we shall see that it is possible to surmount this mismatch problem through proper design of the support tool.

6.1 Generic User Support System Design

From our study of the works of others and from our own investigations, we have found that there are some general principles that apply to the design of operational support tools. For the most part, these stem directly from traditional engineering design practices. Customary design features include flexibility, modularity, incremental growth capability, and independence of modules from the control structures. We follow these features as much as possible. However, the nature of the plant environment guides the design and development of useful operator aids. Next we discuss

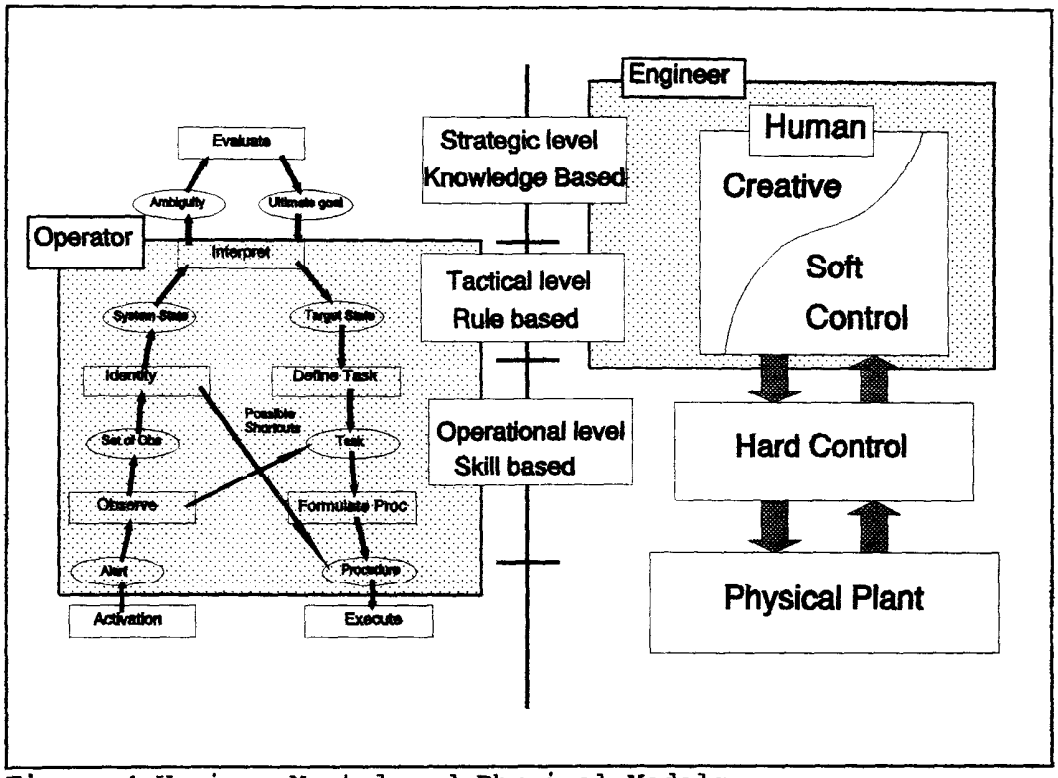


Figure 4 Various Mental and Physical Models.

how the characteristics outlined above manifest themselves in the GUS design in particular.

What single aspect is the most important to capture and hence most influences the design of an operator aid? The distributed architecture stands out as central since this dictates the whole character of the station. It IS, in fact, the physical station. Diverse and disparate knowledge bases and asynchronous activity naturally follow. The plant has been ENGINEERED from the ground up and any aid must acknowledge this fact. This complex system must be controlled and this implies measured data, system knowledge and agents that act in this control function. The distributed and diverse nature of the plant also implies that the controlling agents be diverse. No one aid can ever hope to encompass the needs of the operator. Rather, many small aids need to be developed in a coordinated manner following the natural organization of the physical plant and activities of the operational staff. But as soon as one prescribes separate agents in the design of the operator aid, artificial divisional boundaries have been created and these boundaries must be bridged by message passing of some form. This, in turn, implies a message format and a communication medium. Since the aids may be as physically distributed as the plant, some form of local area network is indicated. There will inevitably be copious quantities of data and it is usually prudent to include a data storage agent which can act as a message coordinator or post office. Such an agent is called a blackboard or, perhaps more correctly, a postboard. By a judicious choice of agents and their duties, message passing can be minimized. Indeed, message passing has proven to be a bottleneck in the past, leading to a design principle of making the agents as proactive, persistent and persevering as possible, meaning that they act with the least instructions, will continue to act until told to stop (or their duties are complete), and will try hard to complete assigned tasks even in the face of incomplete or inconsistent data.

The individual aids, operating in parallel, are of considerable utility by themselves but the true power of the paradigm comes when the various individual aids cooperate in problem solving (concurrent engineering).

We note that this is a conceptual organization. Implementation could vary from single processor,

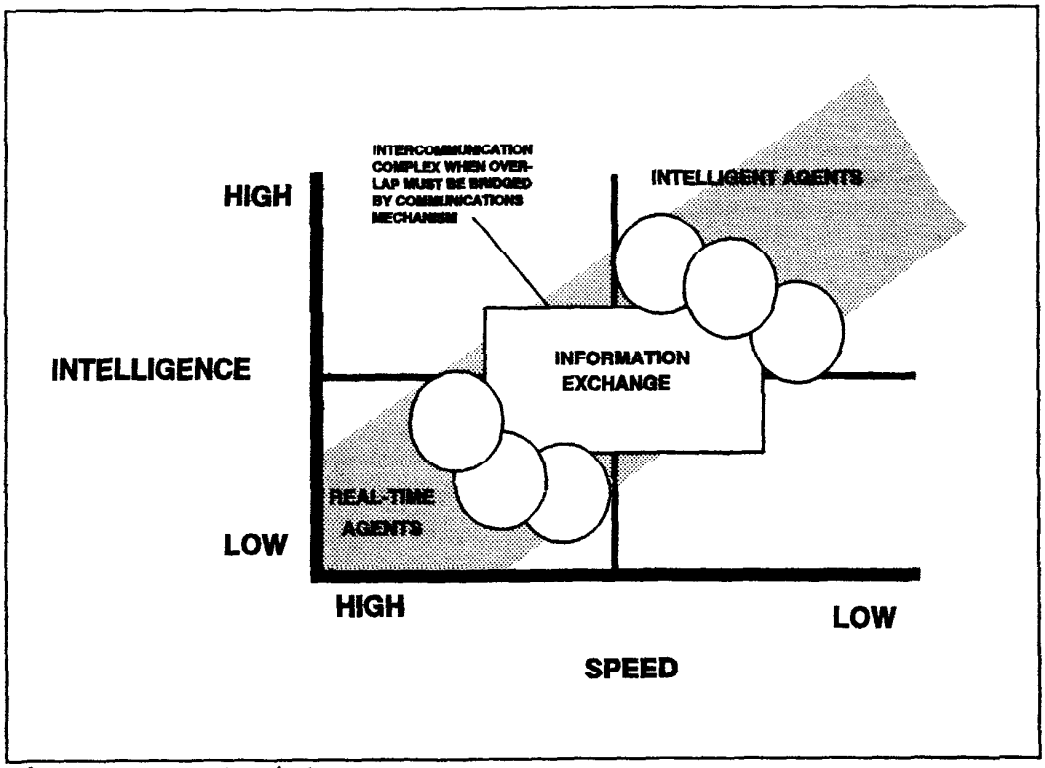


Figure 5 Constraint Space

single tasking to multiprocessor, multitasking. The implementation could be on one machine or distributed over many machines (tightly or loosely coupled).

As mentioned, the plant has been engineered and now it must be operated; but the mental models in the two activities are different, leading to a dilemma for the designer of operator aids. The basis for the plant (and plant models) are different from that of the operator. Interaction with the operator must be on the operator's terms. To make the operator aid more understandable to the user, the aid is developed along anthropomorphic lines of manager, supervisor and technician. Further, the information presented to the user must be consistent with the decision making process of the user. For instance, inevitably, the user is faced with making a choice:

'Of the possible problems facing me, which is the one that I should pursue now?'

This implies pre-enumeration and recognition of this feature simplifies implimentation enormously.

Discussion on the obvious plant characteristic of real-time performance has been postponed until now since, as important a characteristic as it is, it does not play a central role in operator aid design at the level that it is being discussed herein.

Any operator aid of value must contain knowledge of the plant that is up to date and it must reason about that knowledge at an appropriate pace. Thus, there are two time related aspects of a real-time aid. Our solution here is to functionally and temporally abstract the design so that specialized agents can handle the time critical events like data acquisition, filtering, trending, etc. leaving the interpretation and analysis of the data to agents on a higher level. This is precisely how real-time reasoning is handled in the physical plant. Reasoning heuristics provide the methodology for dealing with the diverse and sometimes conflicting implications of plant data and alarms. Real-time for an operator aid means that the aid must be able to provide aid at the human's pace, not the plant's pace. This is so because the plant has been engineered to be so.

A real-time system is one that reaches its conclusion before the real system does. Can the aid

keep up? We need a strategy to cope and one possible strategy is to provide a fast but approximate solution with subsequent refinement later if possible. This is somewhat like the navigational technique of establishing an approximate heading early on and correcting en route.

Reasoning in real-time about real-time data is in its infancy. No one has yet developed truly real-time inferencing and current expert systems are limited to treating time evolution as a series of finite states. Consequently, current operator aid designs should concern themselves with aiding the operator in making real-time decisions by providing the right information at the right time and place in a manner that integrates well with the thought processes of the user. However, nothing in our system precludes the use of operator heuristics (once they are uncovered).

The ideal environment would include a communication highway (like ethernet) to allow peer to peer interaction, involve distributed processing, permit short circuit heuristics (as per Rasmussen), have no implied or enforced control structure (orthogonal to the control plane) and permit parallel and concurrent problem solving. Figure 6 illustrates an implementation that has the required generality and flexibility. No specific control structure is implied; it is simply a fast, inexpensive message passing medium that is available today. Note that incoming plant data is broadcast to all agents but the primary direct users of the plant data are the data acquisition agents and the blackboard. All agents are free to interact with other agents. (the blackboard and the data acquisition agents are just specific instances of agents).

6.2 A Specific Implementation: OPUS

McMaster University is developing an advisor on turbine condenser tube leaks and reactor derating due to secondary side chemistry problems. Research on this project, has been dubbed the Operator / User Support Project (OPUS). One of the project's goals is to demonstrate the utility of the anthropomorphic approach of applying the blackboard paradigm partitioned along the lines of manager - supervisor - technician, allowing symbolic - numeric coupling with the inherent efficiency of asynchronous operation in real-time.

To date, a procedural code has been developed to provide a timing benchmark and to validate the logic. A PC-based multitasking blackboard prototype has been developed, tested and benchmarked for a toy problem [11]. A PC-based blackboard version of the central sampling advisor is currently in beta-testing. To explore the migration of the aid to a distributed architecture, a LAN based on ethernet and TCP/IP between a SUN Sparc Station and 3 486 PC's has been installed at McMaster and a socket based message passing library over the LAN has been established for UNIX-UNIX communication. PC-PC and UNIX-PC socket libraries are under development. Interaction with Pt. Lepreau continues. As has been found in many other knowledge based tool development, the biggest bottleneck is discovering the expert's knowledge and organizing that knowledge in a coherent manner. Plant operators do not have a lot of free time to ruminate for the knowledge engineer's benefit and the process of turning inherent expertise into explicit heuristics is not trivial.

The OPUS system is depicted in Figure 7. Note that the structure of the aid follows the generic principles outlined in this paper. Currently it is not tied directly into any plant data but it could easily be linked to Pt. Lepreau's GATEWAY LAN giving access to existing Chemistry Monitoring System data.

Work has begun on the exploration of the role of an ES as a complementary tool to the current OPUS tool. The rules as built into OPUS have been duplicated as far as possible using CLIPS as the ES shell. Although it is not yet linked to OPUS, it is envisioned that a log file generated by OPUS will serve as input to the ES. The user (and perhaps the regulatory agency) will then be able to review the event history and perform two very important tasks:

1. Ask why.
2. Verify the OPUS logic using a generically different paradigm.

Finally, we have justification for the use of an inference engine for HX related activities.

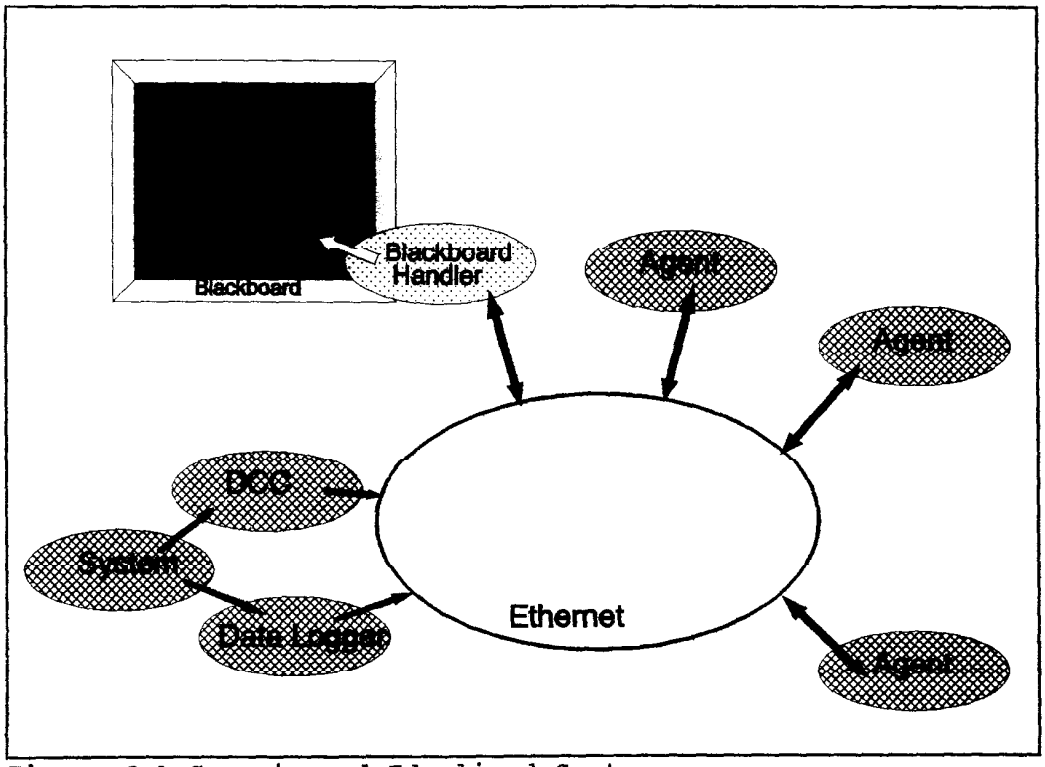


Figure 6 A Generic and Idealized System.

7. CONCLUSIONS

In conclusion, it has been found that the design of KBS aids for HXs follows quite directly from a study of the physical situation and the human experts and users involved. There appears to be nothing inherently difficult in implementing the tools once the knowledge format is cast. The difficult part is the knowledge engineering: the experts have limited time to impart their knowledge to the knowledge engineers and even if adequate time were allocated, the expertise often has to be 'discovered'. These are challenges however, not reasons to default to the status quo. Even if no aid were ever to be implemented, the knowledge discovery would justify all the effort.

8. ACKNOWLEDGEMENTS

This work has been made possible through funding the Natural Sciences and Engineering Research Council of Canada Research Grants and Strategic Grant number STR0118177 and collaborative work with Atomic Energy of Canada. HTFS-Harwell Laboratories in the UK, and the Atomic Energy Control Board of Canada. The McMaster University research group contributing to this work include Dr. W.F.S. Poehlman, A. Bokhari, C. Baetsen, and R.J. Wilson (Engineering and Computing Services).

9. REFERENCES

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

2. 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.

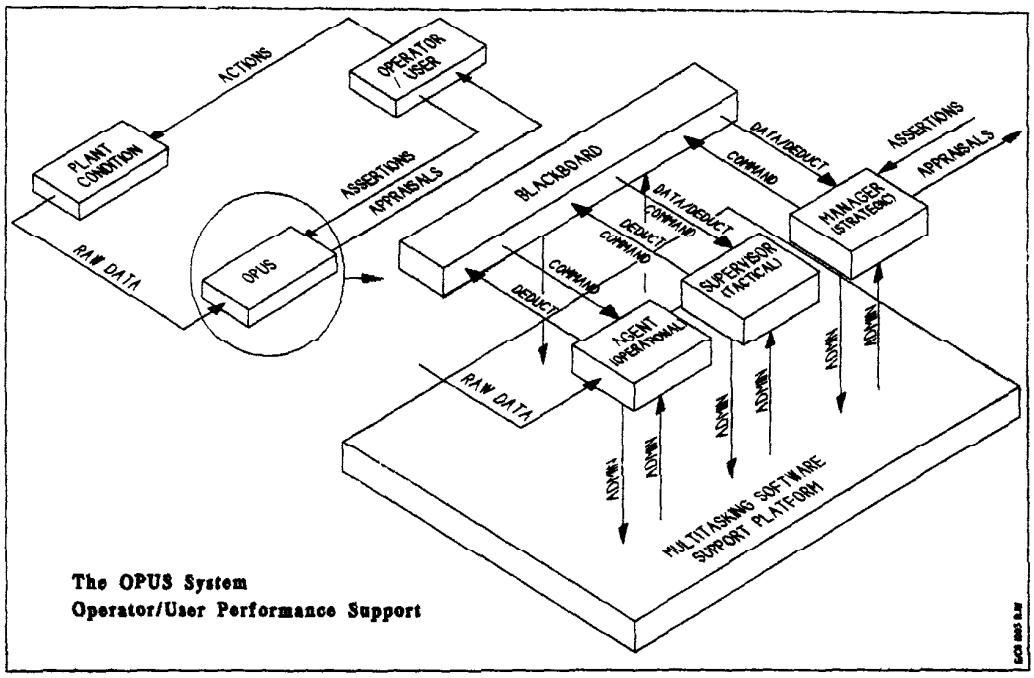


Figure 7 OPUS system overview

3. John A. Bernard, "Issues Regarding The Design and Acceptance of Intelligent Support Systems for Reactor Operators", ICHMT 2nd International Forum on Expert Systems and Computer Simulation in Energy, University of Erlangen, 17-20 March 1992.

4. Jens Rasmussen, "Information Processing and Human-Machine Interaction: An Approach to Cognitive Engineering", North-Holland Series in System Science and Engineering, 1986, ISBN: 0-444-00987-6.

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

6. B. Linnhoff, D.W. Townsend, D. Boland, G.F. Hewitt, B.E.A. Thomas, A.R. Guy and R.H. Marsland, "User Guide on Process Integration for the Efficient Use of Energy", Institute of Chemical Engineering, 1983.

7. "Selection and Costing of Heat Exchangers", ESDU 92013, ISBN 0 85679 818-5, February, 1993.

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

9. C.J. Norman, "Heat Exchanger Selection Part 2: Selection and Preliminary Design of Shell and Tube Heat Exchangers", AERE-R 12202, May, 1986.

10. L.R. Lupton, J.J. Lipsett, R.A. Olmstread and E.C. Davey, "Foundation for Allocating Control Functions to Humans and Machines in Future CANDU NPPs", Proceedings of an International Symposium on Balancing Automation and Human Action in Nuclear Power Plants, sponsored by the IAEA and OECD, Munich, July 9-13, 1990, IAEA-SM-315/28 pp 349-367, also as AECL - 10198.

11. A.S. Mahmoud, Wm.J.Garland and W.F.S. Poehlman, "Multitasking Strategies in Support of

a Knowledge-based Operator Companion", AIENG'92: Engineering Applications of Artificial Intelligence VII, ed. D.E. Grierson, G. Rzevski and R.A. Adey (Elsevier Applied Science, New York: 1992) pp. 1001-1015, Waterloo, Ontario, Canada, 14-17 July, 1992 (Wessex Computational Mechanics Institute, U.K.)