1-1-1986

The role of expert systems technology in design

Kristian Lien
Carnegie Mellon University

Go Suzuki

Arthur W. Westerberg

Carnegie Mellon University.Engineering Design Research Center.

The Role Of Expert Systems Technology In Design

by

K. Lien, G. Suzuki, A. W. Westerberg

EDRC-06-13-86

September 1986

THE ROLE OF EXPERT SYSTEMS TECHNOLOGY IN DESIGN

by

Kristian Lien*
Go Suzuki*
Arthur W. Westerberg

Department of Chemical Engineering and the
Design Research Center
Carnegie Mellon University
Pittsburgh, PA 15213

April, 1986

ABSTRACT

Using a scenario -format, this paper first reviews the nature of chemical process design, showing that designers quickly make major **decisions with** minimal information and constantly revise their strategy to solve a problem. To automate this activity on a **computer will** require models of the process being created at several levels of abstraction as well as models that capture the beliefs of the modeler about the abilities of himself, others and the aids available and models of strategies for complex pi-obiem solving.

The second section of the paper extensively reviews *c^rr^-z* expert system concepts, illustrating each of them with *^.^--.inn* examples. **We argue that** expert systems *^r&* knowledge based. u*e describe many **of** the control strategies used in today's L, v^+-.; .-. and also *consider* different problem representations -- r-. i^-r. logic **and frames** - and indicate when each might be'*pr&i&mrd. H* last **section gives** our views on what will be involved in creating a future expert system *-far* design.

---

1. Norwegian Institute of Technology, Trondheim, Norway

2. Toyo Engineering Co., Tokyo, Japan

INTRODUCTION

This is a paper about expert systems in design. Their role is not yet established, so this paper is prospective rather than retrospective; we do not describe the highlights and shortcomings of existing systems but rather attempt an analytic approach to the -following questions.

- What is the nature of design?
- Why *Are* experienced designers good at it?
- How *And* to what extent can computers *da* it?

Design is a complex, open-ended collaborative activity- It we could characterize it completely, we could probably also write an algorithm for it. We cannot, so we have to look to how design is performed by experienced designers, trying to understand how they solve design problems. The obvious explanation as to why experienced designers *Are* good at design is that they have experience on which to rely, a basic understanding of the concepts involved *in* design and intuition to guide them in the solution process.

We will argue that these reasons can be broken down as follows.

- They have models *oi* chemical processes on multiple levels and *Are* able to relate these models to each other.
- They *Are* able to formulate strategies *-for* attacking the problem (although often unconciously) .
- They *Are* able *to* switch between models and reformulate strategies as a response to the preliminary results of a partially completed design.
- They *Are* able to accumulate knowledge, both factual and strategic, along the way.

In the absense of theories of how design is done, the only way computers can be utilized to do design is by implementing models *o-f* how experienced designers do it, using the same information in a similar way. To the extent we can emulate designers, computerized models of design may contribute both to a better understanding and an improvement of the design process.

The kind of expert systems we are outlining above will probably continue to be research efforts for a long time, since the strategies used by human designers are poorly understood as yet. Industry is, however, already actively looking at the use of expert systems in design, often with a suspicious attitude. In the context of more limited problems within design, where the number of concepts involved is small and the flexibility of human strategies for solving the problem is limited, a number of successful expert systems are likely to be produced in the near future. They will be created because expert systems are able to process qualitative, heuristic and causal information, which seems to be abundant in design problems.

CREATING EXPERT SYSTEMS IS MODEL BUILDING

As human beings we are able to make observations about the world around us, and we are able to interpret these observations so that we can maintain a set of beliefs about "how the world works." We shall call these beliefs our model of the world: models are sets of beliefs which create expectations about the behavior of the world, guiding us in our interaction with it. It follows from this definition that we believe these models are predictive in its nature.

The purpose of having and using models of a system is to reduce the amount of experimentation needed to discover how the system works. Models, the sets of beliefs, are dynamic. As we observe and interpret the nature of the system, our sets of beliefs are extended and revised, so that new expectations are created to guide our interaction with the system. This activity is what we commonly term learning.

In design the system consists of the problem being studied, the organization studying the problem and the way the organization approaches and solves it. We do therefore believe that three types of models exist in design:

- Models of the problem. These models are sets of beliefs describing how we expect physical entities (e.g., processing equipment) and phenomena (e.g., heat and mass transfer) to behave.

- Models of the designers and of the available aids. These models are sets of beliefs describing the expected capabilities and limits of the design team, each individual designer and the aids to be used.

- Models of strategies. These models are sets of beliefs describing how we expect to solve the problem, given the

previous two types of models.

Complex systems give rise to complex models *if* all aspects of the system *are* considered simultaneously in full detail. In *order* to handle complex systems, humans seem to decompose their models of the system, giving rise to models of the system which *provide* different views and different levels *of* detail and *accuracy,* Decomposing the system in this way, designers extract the essential information for different aspects of the system, get an overview *oi* the system's gross behavior and/or analyze parts of the system in greater detail.

Finally it is interesting to note that the above is a model. It is a set of beliefs about what kind of models we need to solve design problems.

Computer-aided design has until now been mainly concerned with creating well defined models within a mathematical framework, leaving the parts of the design not captured by quantitative methods to the human designer. Furthermore, the main focus of these aids has been on the first type of models listed above, those which, characterize the behavior of physical items and phenomena. However, a new approach to computer—aided design is now emerging, *an* approach where the computer is also supposed to handle the more qualitative aspects of design: the expert systems approach.

Considerable objections *have* been *raised* against the *id&3* that computers can do design, often based on the following* "observations[11] about current expert systems.

- They do not represent anything new. They *are* simpl/ implementations *oi old* technology in new, *-fancy* programming environments.

- They have no basic understanding of the basic concepts, b_t chain through a number of more or less reliable rules as idiot savants.

- Etc.

THESE ARE NOT THE ISSUES!

The question of whether expert systems can be useful or *not.* is basically a question of whether we believe that a diversity of models of the three types mentioned above can be implemented in a computer program and the extent to which we appreciate that this is an activity that should be attempted.

A DESIGN SCENARIO

The *Scenario*

We present in this section an example of a designer attacking the preliminary design of a chemical process to manufacture ethyl benzene from benzene and ethylene feedstocks. Ethyl benzene is a precursor to the manufacture of styrene- We shall be looking over the designer's shoulder to watch how he moves quickly to a first solution of this problem. Our purpose is to show that he formulates a strategy first to attack the problem, that he uses models at various levels of detail and that he constantly replans his approach.

Our designer starts by taking a quick look at the problem statement, which indicates roughly the available raw materials, the likely reaction pathway and the desired product. He decides first to focus in on the *reactor*. The main reaction step is

$$C_6H_6 + CH_2{=}CH_2 \rightarrow C_6H_5\text{-}CH_2CH_3$$

Benzene + Ethylene —> Ethyl benzene

He surveys the literature and/or experimental results to gain an understanding of the reaction conditions. The reaction is found to be exothermic and can be run as a Friedel-Crafts reaction in the liquid phase or by a carboniurn ion-like mechanism *over* a-zeolite catalyst in the vapor phase. In both options the *reactor* feed contains benzene in significant excess so that essentially all the ethylene is converted and does not have to be recycled. Both ^re used as the bases for existing industrial processes.

Our designer *decider* he will first attempt to select the phase in which to run the reaction. It appears to be the crucial decision for this process.

. In the liquid phase reaction, the catalyst cannot tolerate water so the feeds must be thoroughly "dried." Also the catalvst will likely be washed from the reactor effluent using water, and it will form a very corrosive mixture that will require special materials of construction. Reactor conditions will be about 5 to 10 atmospheres and between 150 to 180 °C. Selectivity is high but conversion low so a large recycle will result. The literature suggests a benzene recycle ratio as high as 0.5 to 1.0.

Reaction in the vapor phase also must be water free. Pressures will be about 15 to 30 atmospheres and temperature around 400 to 450^{o}C. Catalyst coking will occur requiring that a spare reactor exist so one is being regenerated while the other is in use. Conversions are much lower than in the liquid case, giving reported benzene recycle ratios as high as 5 to 20. Selectivity is high.

The ethylene feedstock is available at a high enough pressure to supply the vapor option without the need of a feed compressor; however, the vapor option requires a much higher recycle ratio. Operating at higher temperatures, it may offer more heat integration possibilities. Our designer is unable to decide between the two alternatives and puts that decision on hold until he gathers more information and insight into the process.

He next settles on a set of reasonable specifications for the products and the available feedstocks. Using these specifications he uses a broad task oriented process description to establish first guesses at the material balances and therefore how much of the feedstocks will be needed and/or how much of the products will be produced for each of the alternative. Fig. 1 illustrates the very abstract nature of the description used.

He next lists the obvious needs for the process for each of the alternatives. A drying system for the benzene feed is needed as the benzene feed contains a small amount of water. There will likely be the need of a small purge for the recycle to avoid the buildup of minor contaminants.

He checks on the available utilities; here low, medium and high pressure steam, cooling water and electricity are assumed to be available.

He guesses the following technologies which will be used. For the liquid phase option the reaction will be carried out in a homogeneous phase with continuous catalyst feeding to be followed immediately by a water wash to remove the catalyst. For the vapor phase, the reactor will likely be a fixed bed reactor. Drying of the benzene will require either distillation or adsorption; it is not obvious which to select so he also defers this decision. Nominal values are selected for the temperatures and pressures in each section. He sketches his ideas, Figs. 2a and b.

Again without numbers being calculated, he conjectures the possibilities for heat integration in each alternative, and adjusts his estimate of the needed utilities. He is still only thinking in terms of "large," "medium," and "small." Both cases appear to have possibilities for heat integration, but the vapor phase seems to offer more because the higher temperature heat from the reactor can provide more of the heat needed by the

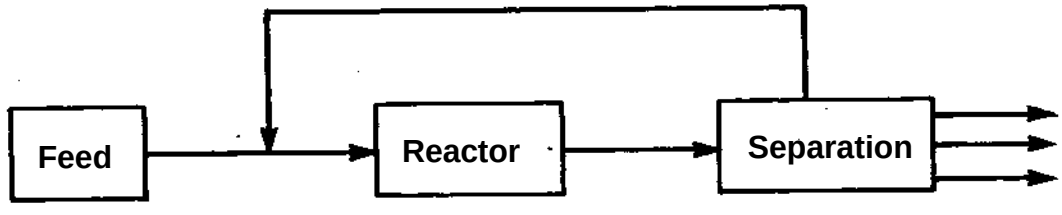
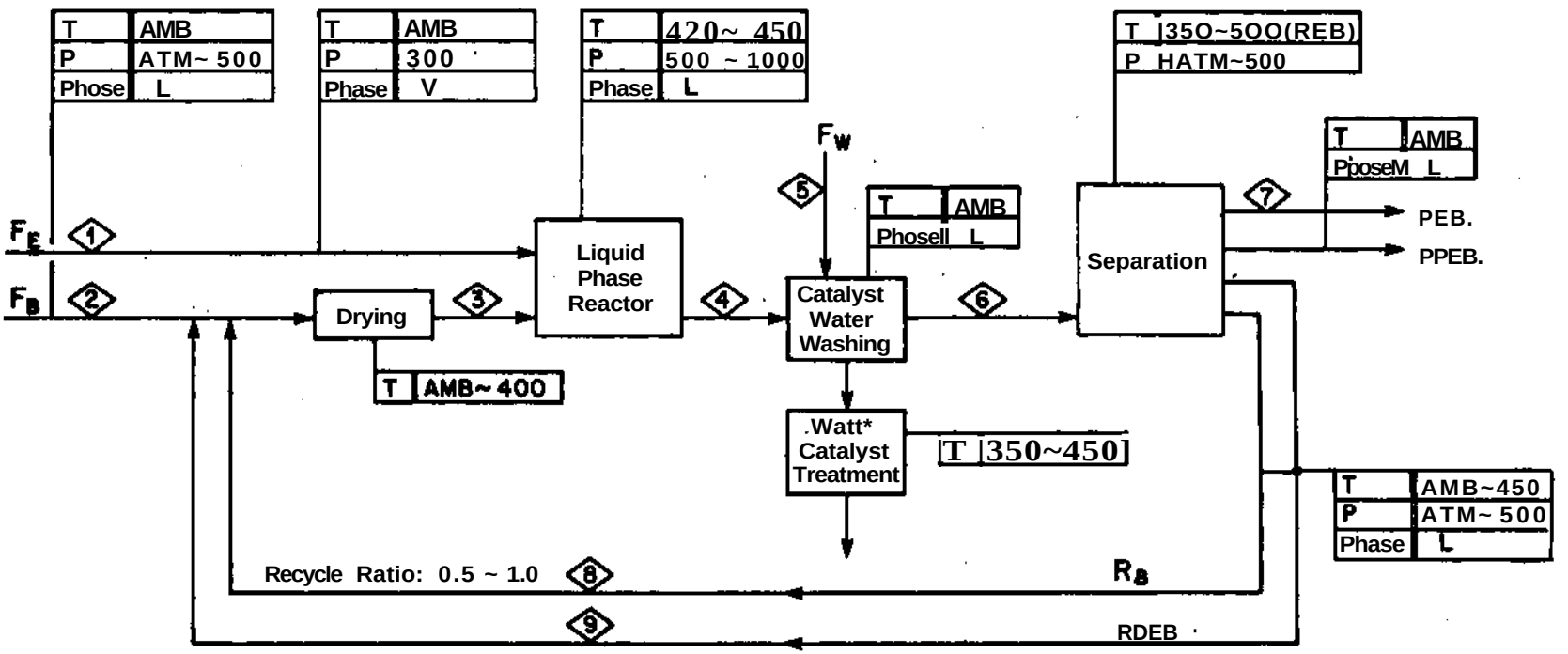


Fig. 1 Simple Block *Diagram* for Process



STREAM MATERIAL BALANCE

AMOUNT / COMP.	①	②	③	④	⑤	⑥	⑦	⑧	⑨
E	Med	-	-	-	-	-	-	-	-
B	-	Med	Med	Med	-	Med	Sml	Med	-
EB	-	-	-	Med	-	Med	Med	-	-
DEB	-	-	-	Sml	-	Sml	-	-	Sml
PEB	-	-	-	Sml	-	Sml	-	-	-
CAT	-	-	-	Soill	-	-	-	-	-
W	-	Sml	-	-	Med	Sml	-	Sml	Sml
TOTAL	Med	Med	Med	Med	Med	Med	Med	Med	Sml

Med: Medium Amount
Sml: ftmnll Amount

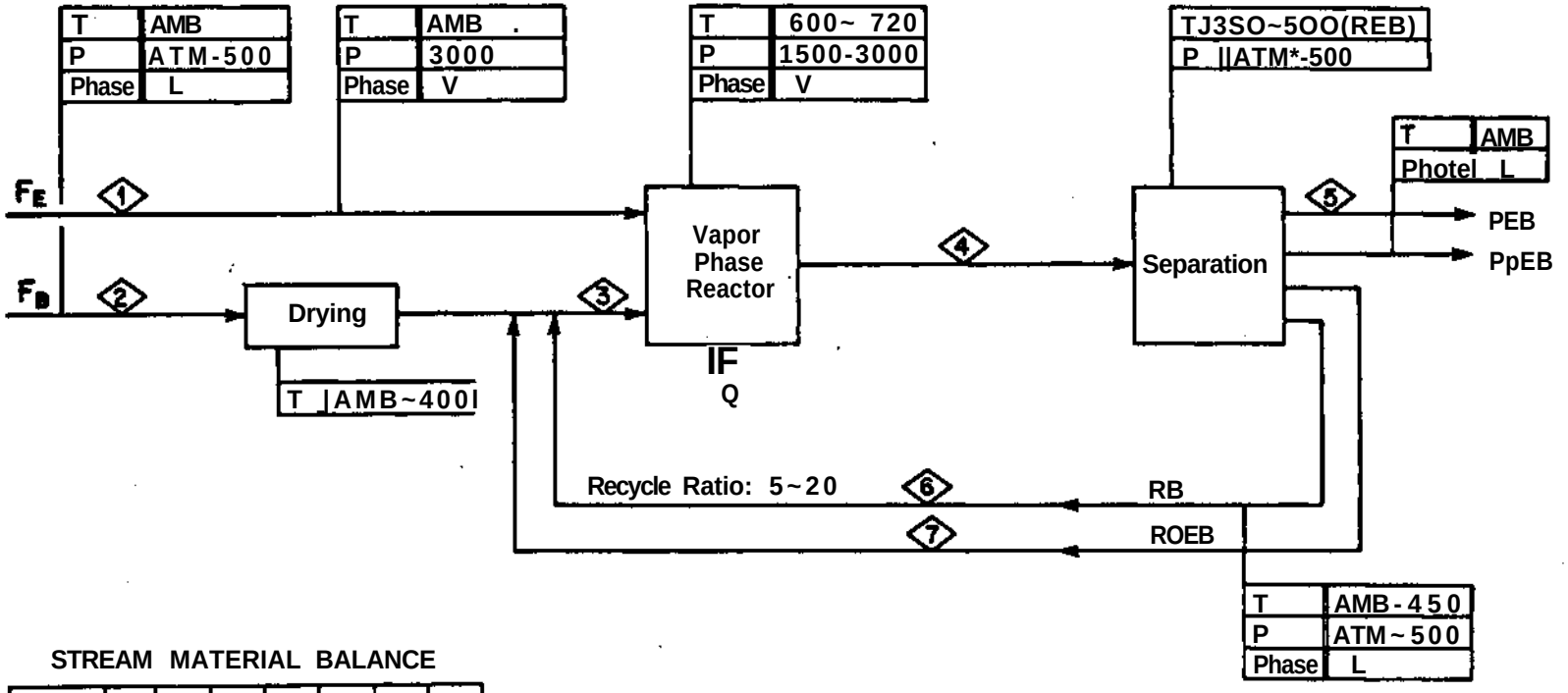
STREAM CONDITIONS

TOO	
P(KPa)	
Phase	

LEGEND

F : Feed
P : Product
R : Recycle
E : Ethylene
B : Benzene
EB : Ethylbenzene
DEB : Dlethylbenzene
PEB : Polyethylbenztne
CAT : Cotolyst
W : Water

f.ig „ :;:a *hi*ock I-1ow I)iagram *ol* L.iqui*d* Phasie Reacti*an Process*



STREAM MATERIAL BALANCE

AMOUNT COMP.	O	<1>	<§>	<4>	<|>	<!>	<7>
E	Mtd			-		-	—
B	—	Med	Lge	Lgt	Sml	Lge	—
EB	-	-	-	Med	Med	—	—
DEB	-	-	Sml	Sml	Sml	—	Sml
PEB	-	-	—	Sml	—	—	—
W	-	Sml	—	—	—	—	-
TOTAL	Med	Med	Lge	Lge	Med	Lge	Sml

Med: Medium Amount
Sml : Small Amount
Lge: Large Amount

Fig. 3.1 Material Balance Diagram of Vapor Phase Reaction Process

separation section of the process (which he is assuming will be predominantly based on distillation). He also notes, however, that the larger recycle for the vapor option will require more separation and thus more heat to operate the separation section. He envisions the possibility of cascading the heat from the reactor through one column into another by appropriately selecting the pressure levels for them. From his estimate of the temperature differences across the possible columns, there appears to be enough driving force to allow this type of option.

He decides next to make a rough estimate of the capital costs for each option, concentrating on the reaction section. He assumes this section will be relatively expensive because of the corrosion problem in the liquid phase option, the spare reactor required for the vapor phase option and the cost of catalyst. The separation costs will be dominated by distillation of the benzene from ethyl benzene in the reactor product, a separation that does not look too difficult. He assumes the drying section will not be too expensive.

To gain an appreciation of the separation tasks which are needed, he decides next to establish a base case for examination. In this case he sees little reason not to consider the "direct" sequence for the separation tasks required (the direct sequence is where the most volative component is removed one at a time in each consecutive column). He still has his eye on the use of heat integration.

He plays first with the separation scheme for the liquid reaction phase option to determine the potential for heat integration. He now considers each section in more detail. First he notes that the benzene "drying" section can be done using azeotropic distillation. He notes that both this section and the catalyst removal section can operate at temperatures low enough to use low pressure steam. He also discovers that the reactor will be unable to supply the highest level of heat desired for the distillation system. Fig. 3 illustrates his analysis. His analysis is based on assumed relative magnitudes for the columns - will they have large feeds and are they easy separations (based on looking at the boiling points of the species involved).

This diagram represents each column as a higher temperature heat sink (the heat required for the reboiler) and a lower temperature heat source (the heat which is expelled by the condenser). He knows these heats will be approximately equal to each other. Thus each column is represented by a box on this diagram. The benzene/water column, for example, will need heat somewhere in the range of the high pressure steam utility and will expell its heat perhaps at the level of the low pressure steam utility. It is of medium difficulty and thus will have a medium drop in temperature across it from its reboiler to its condenser. The column to remove the diethylbenzene (of which a

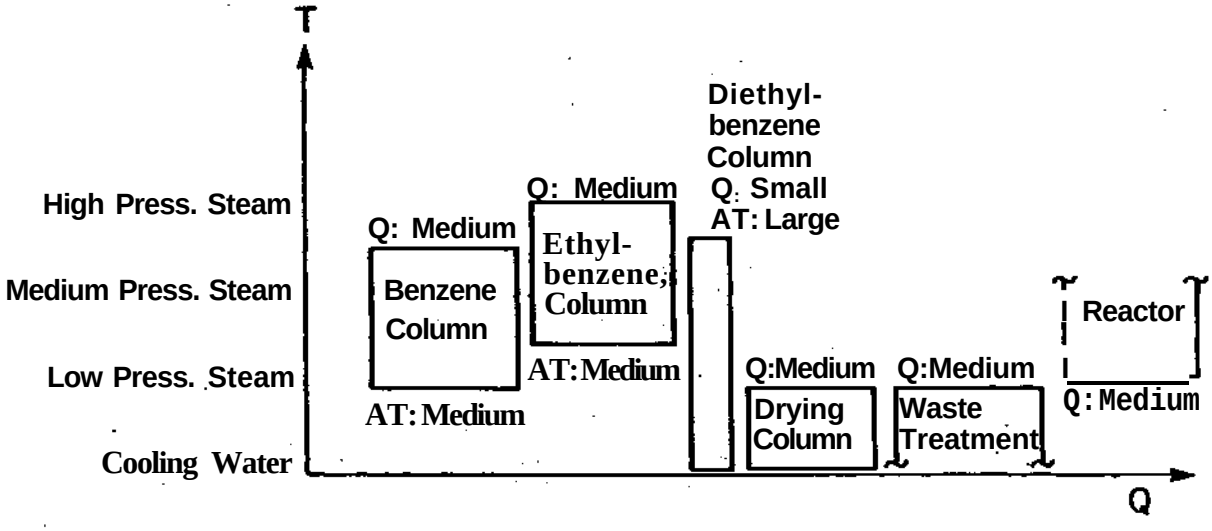


Fig. 3 Qualitative Assessment *of* Heat Loads *-fat-Liquid* Phase Reaction Alternative

small amount will -form in the reactor) is easy to separate from ethylbenzene. This column will use little heat but will have a large temperature drop. To operate it at reasonable temperatures it may be necessary to run it under vacuum condition, an option he would prefer to avoid. The *reactor* is shown on the right. It is not hot enough to supply the heat needed -for most of the colLimns.

Fig. 4 shows a similar sketch he generates *-for* the vapor phase reaction option. Using rough numbers our designer notes that the heat *-from* the *rGSLctor* (shown on the right) is able to supply only part of the heat needed by the columns. To supply the highest temperatures, he wonders *ii* there could be a coking problem for the reactor.

We stop watching the design being *created* at this point as the essential ideas *^r&* now exposed.

Observations about the *Scenario*

The scenario illustrates several points we wish to make about the design process. First the designer uses several kinds *a-t* models, both of strategies for solving and of the process and the equipment in it as the structure is being *cre&ted.*

At the highest level he followed the strategy to look first at the reaction, then create the overall block structure (Fig. 1), next develop details of each and look at the heat integration. At a lower level he predicted the characteristics of the reactor, the separation subsection and the details of the heat integration. Some general strategies were to try to select one good alternative among the many possible, but, if / that proved impossible, to defer the decision making until more information had been generated. Holding back on choosing whether to use a liquid or a vapor phase reactor is a case in point.

Different levels of system models were used throughout as needed. The overall system was modeled first by a simple block structure. Later the blocks were expanded into a more detailed description. He used unit operation models of different complexity; e.g., at first he modeled a distillation column by guessing only its top and bottom temperatures at one atmosphere (to gain an idea of the temperature drop to be expected no matter the temperature at which is it to operated) and that it would have a small, medium or large feed flow into it. In later steps he will use first shortcut models and finally rigorous plate-to—plate models for the columns finally selected to assure they will perform as desired.

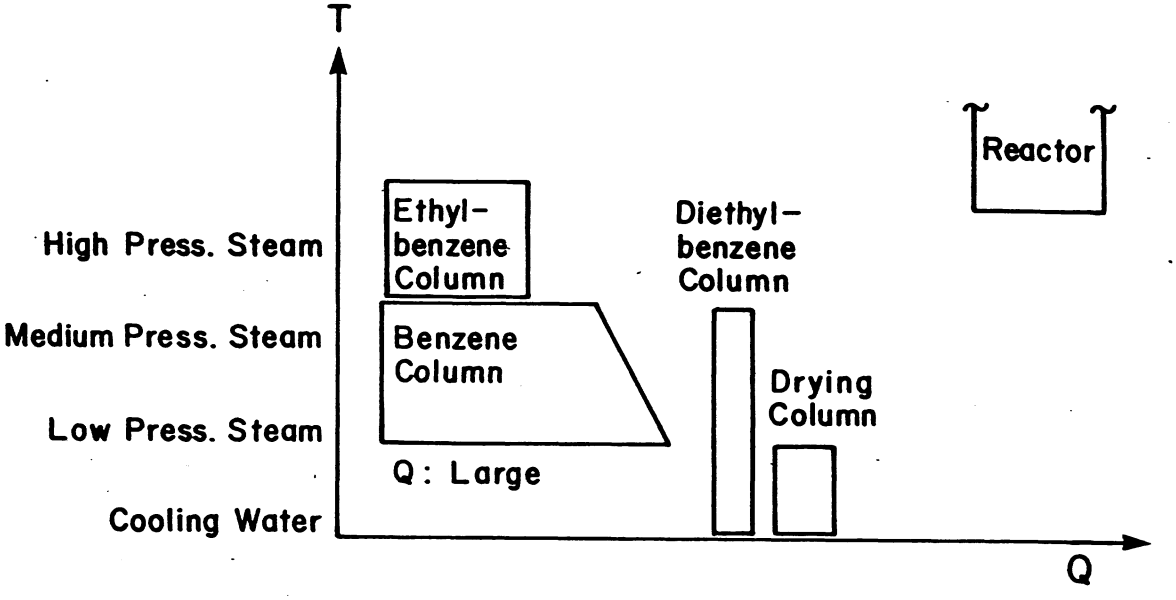


Fig. 4 Qualitative Assessment of Heat Loads for Vapor Phase Reation Alternative

Notes: Reaction generates less heat in this alternative because it's inlet must be vapor. Drying Column is also smaller because the recycle need not be dried. Benzene column is drawn for vapor feed which might be a good idea because the reaction is in the vapor phase.

Our designer moved from higher level to lower level strategies and back as needed; he moved from less to more detailed models of the process and the unit operations in it, again as needed. He knew the general behavior and tendencies of the equipment. For example he knew that by operating a column at higher pressure, it will operate at hotter temperatures *(and* he may have known that it will probably also require both more heat and a larger temperature drop to accomplish the same separation task).

He could guess which parts of the process calculations he should postpone -*for* the moment and where to get more *accurate* information. He was continually assessing the results and modifying his approach in response to them.

THE NEED FOR MODELS

As described in the introduction we will need several kinds of models if we ever hope to create a computer system that can perform competent design. We shall use this section to describe the three types of models that we stated *are* needed: of physical things, of the designer and available aids and of the strategies to be used. We claim that we cannot program a computer to solve design problems for which we *are* unable to establish these models.

MODELS OF PHYSICAL THINGS

In general models of physical things *are* quantitative models which may exhibit varying degrees of *accuracy.* At one extreme they *are* in terms of large sets *oi* "algebraic,[11] *ordinary* differential and/or partial differential equations. They *may* also include the use of discrete variables which can, for example, indicate the existence *or* nonexistence of parts *o+* the structure that may be in a model.

At the other extreme these models too may be qualitative and be expressed only in terms of large, medium *or* small flows, tor- example. They may also be in terms of the tendencies o-f the models, such as increasing the pressure will *increase* the temperature level at which the entire distillation column operates.

MODELS OF THE DESIGNER AND AVAILABLE DESIGN AIDS

To attack a design problem, the designer must also have a set *ai* beliefs about his own capabilites and *of* the capabilities of others with whom he may need to work. He has to know that he can *or* cannot solve certain classes of problems or that others in the group can *or* cannot help- If he has no model of either himself *or* of others, he must then have a set of beliefs that he can discover this information in time to use it for the problem at hand.

He must also have a set *of* beliefs about the capabilities of the available design aids which he and others will be using to solve the design problem. Either that or he has to believe that he can find out about them in time to use them.

Without these beliefs, he cannot start to formulate strategies to solve a design problem. These beliefs *are* his models of himself, others and of the design aids.

MODELS OF STRATEGIES

If we wish to automate the solving of design problems, then the third kind of modeling required is that of the strategies by which problems can be solved. Indeed this type of modeling is one of the major issues involved with much of the research in the area of expert systems. /

When an experienced designer performs a design, he makes rnanv significant decisions quickly and with virtually no computations. What *are* the representations he is using to visualize the problem and its solution, and how is he using thpin to discover the next key decisions. Are these the representations and strategies which *are* best for solving the given problems?

In this section we shall illustrate issues which relate to strategy. Many examples of variations in strategy exist- Some attempt to mimic the steps taken by an experienced designer; others use more generic problem solving concepts. .

To illustrate alternative strategies, we review four which have been proposed to invent the structure for complete flowsheets: the strategies used by (1) the AIDES program (Siirolla, et al

C19713, (2) the BALTAZAR program (Mahalec and Motard C19773), (3) Douglas C19853 and (4) Grossmann and coworkers C19833. Each of these approaches is to invent a *chemical* process given the desired products, the available raw materials and the allowable *reaction*(s).

The AIDES program assumes the reactions *and* the details about their conversions, temperatures, pressures *are* specified by the user -*for* the process to be invented. Assuming any unreacted raw materials will be recovered and recycled and thus that raw materials *are* totally consumed and using market prices *for* the raw materials and products, AIDES selects the amounts to use or create *o-f* each by solving a small linear programming model which maximizes gross profit - i.e., the difference between the income *received -from* the sale of products and the costs of the raw materials.

Next all possible distinations for species *are* coupled with all possible sources for them. *For* example, the reactor inputs and the products *are* identified as destinations for species; available raw materials and reactor outputs *are* identified as sources. If a species A exists *in a* source and a destination, the two *are* coupled. For each coupling AIDES develops a *score* that represents the desirability of actually allowing the source to supply the needs for species A for the destination- The scores form the basis of a linear objective function to be maximized, subject to the *linear* material balances and constraints written using molar flows for each *o-f* the species.

If different species *are in* the same source and *are* found to supply different destinations, AIDES proposes separation task *to* split the species in the source mixture and then converts these separation tasks into actual separation processes using heuristics. Finally heating and pressure changing tasks *zre discovered as needed.*

One can see then that the strategy has been to break the overall design into a hierarchy of steps, each solved completely before carrying out the next: reaction selection (given), selection of raw material and product amounts, allocation of species, selection of separation tasks and finally heating, cooling and pressure changes. No iteration is performed in finding the solution.

BALTAZAR has a quite different strategy to find a design. It posts a goal to produce one of the products. If there *are species* in the product not available from any of the raw materials, it looks for a reaction to produce that material. Reactor inlets become new goals, outlets become sources. BALTAZAR works essentially on one product at a time, creating the structure backwards from the product through reactions to the raw materials until the structure can produce it. Along the way it develops the need for separation tasks, mixing and so forth. It

also considers any of the sources which can be scaled up in the previous parts of the solution to be available as sources. The final step it considers is to include heating and cooling tasks. Because the solutions reflects the order in which the original goals were considered, the system removes structure and changes the order of the goals, iterating until the structure does not change. It also looks at the tasks produced and removes redundant structure.

Douglas' approach is to make decisions based on heuristics. Of the four strategies, it most closely mimics the steps taken by an design engineer. The first set of heuristics create the input-output structure of the flowsheet. These decisions select the feeds to use, the products and byproducts, whether there will be a purge, constraints on its size, and so forth. The next level of heuristics establish the recycle structure of the flowsheet and the reactor configuration. Separation system synthesis occurs next and finally heat exchanger network synthesis.

Grossmann's approach is to invent first a superstructure in which are imbedded the many alternatives he wishes to consider. A significant part of the research to support this approach is the invention of the form of the superstructure for each of the problem types he has considered. These structures are the result of careful consideration of the problem and the desire to minimize the number of discrete variables required to formulate the optimization step to follow. For the superstructure, Grossmann builds a mixed integer linear programming (MILP) model and solves it using existing codes. More recent work allows the use of mixed integer nonlinear programming models.

It should be clear that there are many different stategies and representations for setting up and solving these design problems. It is not yet clear which is the best approach, if indeed any one is.

It would seem that there are several desirable characteristics one would want in a strategy to design processes. First one should want the approach to make obvious decisions quickly. An experienced design engineer will. An example is to choose to use distillation if a mixture to be separated comprises normal hydrocarbons with no boiling points within 25 K of each other.

A second characteristic should be to understand a problem well enough to be able to select which computation should be performed next. Here one is talking about a strategy that is looking at and altering its own strategy as the solution progresses. A simple example in solving the equations for a flash unit can illustrate. The strategy can be to estimate if the mixture is likely to be two phases by examining component vapor pressures. If the vapor pressures are both above and below the flash pressure, the two phase computations are started. The flash

calculation is iterative and it may move across the boundary of the single phase region. This movement suggests the mixture is really single phase, and the program could be directed to compute the bubble point *if* a liquid phase is suspected or the dew point *if* conditions suggest an all vapor phase mixture. The computations to determine either a bubble point or a dew point *Sire* equal in complexity to those -For doing a two phase flash computation and should be *avoided* if not needed.

This simple example illustrates the *ide^ of* a strategy that modifies itself as the solution progresses. For design, however, the problem is infinitely more *complex* than the solving of a flash computation. How does one develop a computer system that can understand the problem well enough at each step to guess where best to do the next analysis?

Another characteristic that seems desirable is to know when decisions matter and when they do not. Often "satisficing" is good enough to produce a solution. Suppose the costs for the process *3ire* found to be for the compressor and that the heat exchanger network can save almost no energy nor can it cost much relative to the compressor. Then the experienced design engineer would simply discover an adequate solution and would certainly not look for the best. He would believe he could only save a few thousand dollars relative to the millions involved in the compressor. How *ca.n* a program know when to satisfice rather than search *for* the problem it is solving? What problem representation will the engineer use to discover this characteristic to his problem?

An example of the strategic use of representation and using satisficing for part of the solution occurs with the following problem. Suppose ones goal is to drive from ones home in New York City to a relative's home in San Diego. *No* one would ever *consider* gathering up all the detailed city street and backcountry road maps to find the solution. The only map needed at first is of the interstate highway system, with the details of other routes suppressed. Only getting to and from the interstate system when starting and finishing the trip requires *mare* detailed maps. The solution of these latter "search" problems heed not be done optimally as little time is involved in their solution no matter how they *3Lre* done, relative to the time *for* the entire trip.

If one is in the early stages *of* a design and is exploring quite different concepts on which to base a flowsheet, then perhaps only bounds will be needed on the costs of parts of the solutions. If an upper bound on one alternative is readily shown to be much less than a lower bound on another, clearly the latter can be ruled out as an approach. Upper and lower bounds *^re* sometimes not that difficult to estimate. An example occurs *in* the design of a heat exchanger network for a design. As Hohmann C1971D shows it is possible to establish a lower bound on the

cost of the utilities required when designing a network of heat exchangers for a set of process streams which *are* to be heated and cooled from specified inlet to specified outlet temperatures. The computation is simple to perform and does not require the invention of a network of heat exchangers to accomplish it. An upper bound is *oi* course the utility use associated with any proposed heat exchanger network.

Another form of strategy is to solve a problem, not to get an answer but rather to gain understanding of the nature of" the problem. A design engineer will almost always do this. What specifically could he be obtaining from this approach? First he could be locating the important problem constraints that were not at first apparent. For example he may discover that the available utilities will not allow high pressures to be used in the process. He will also see that the expensive part of the process is in the separation system for the products, leading him to wonder if he should not be exploring other *reaction* schemes-

There *Are* other questions of strategy that will impact the solving of complex problems.

Frequently when solving a design problem, one will discover that an earlier decisions is suspect. We have the alternative of stopping and returning to and changing the *earlier* decision *or* of continuing along with the design even though we know it is not *correct.* Continuing may be justified if we *Are* in the preliminary stages of the design process and *Are* looking *for* insights about the design and not the final solution. Therefore, we may choose to continue, finding that there *Are* even more *of* the earlier decisions which we may not like. We finally stop exploring the design when we can go no further *or* when we believe the *earlier* errors *Are* making our continuing unproductive. At this point we can assess everything we have discovered, looking at all the partial results and the suspect *earlier* decisions together. With this broader look, we revise some /of them and perhaps revise our strategy to continue the solution process.

HIERARCHICAL STRUCTURE OF MODELS

We have been looking at models in the last three sections. The first were the models of physical things typically in the *-form of* "algebraic[11] and ordinary and partial differential equations. The second were the beliefs (models) that the designer has about the capabilities of himself, of others with whom he is working and of the available aids they might use to create a design. The third considered strategies for solving design problems.

Implicit in many of these models is the notion of decomposition. Large problems must be decomposed to be solved. If one does not decompose such a problem, then simple statistical arguments (Simon, 1969) demonstrate clearly that they will be almost impossible to solve. The companion of decomposition is integration. Most decompositions tear a problem into parts that still interact. To put the parts back together requires iteration of some sort to account for the interactions.

We have posed the need for three kinds of models. Each can be decomposed. We can decompose the models for physical things by-examining the structure for their equations. Partitioning and precedence ordering, two level optimization, and so forth *Are* concepts that arise from decompositions of this type.

We *CAD* decompose the models the designer has of himself into higher level beliefs and lower level ones. *For* example at a higher level, he may believe he *CAR* design azeotropic distillation systems. At a lower level he may believe his office mate can run a computer program to estimate the parameters to solve an azeotropic distillation column computation approximate!y.

The generation of alternative strategies for solving the design problem is the study of decompositions for it.

In mathematical problems, one form *o-f* this type *o-f* decomposition is termed projection, and it illustrates the need for the integration step. Suppose we *Are* solving the following optimization prob1 em:

$$\text{Min} \{ f_0(x,y) \mid f_i(x,y) = 0 \}.$$

We could solve it by the following approach.

$$\text{Min}_x \{ \text{Min}_y [f_0(x,y) \mid f_i(x,y) = 0] \}.$$

The problem is decomposed into an inner and *AH* outer optimization problem. For each set of values selected for variables x, the values for y(x) *Are* found by optimizing the objective only over y. The outer loop adjustments *ai x* and resolving for y(x) *Are* the coordination steps that reintegrate the problem.

The decomposition is useful *ii^* for fixed values of the variables x, there exists an inner problem in variables y which is much easier to solve. For example suppose that, when variables x are fixed, we *Are* left with a set of separated problems in variables y, each *o-f* which is quite small and easy to

solve. We could also have a problem where the inner problem is linear if variables x are fixed. We would be finding the solution by solving a sequence of linear programs. A third alternative is that variables x may only be allowed to take on integer values, whereas variables y are allowed continuous values which can be found rapidly using second order optimization codes.

(In this form of decomposition, one can encounter difficulty if the selected values for variables x leave the inner problem infeasible.)

Suppose we choose to solve a design problem in two levels. At the outer level we select the structure of the input and output flows, in the manner that Douglas constructs a complete flowsheet as described above. One of his inner problems is to discover the separation processes to support the outer decisions. The inner problem interfaces the outer with both material and heat flows, receiving and returning both. Clearly the interface between the two problems must be complete enough to account for both heat and material flows and the decisions of the outer loop must be iterated to find the optimum solution to the overall problem. We get an interesting message here: solving the problem with an inner problem can be a mathematically correct formulation IF the outer loop is iterated and IF the interface between the two problems is correct.

There are other decompositions used in solving design problems. Often the problem is solved by attacking it at different levels of abstraction. At first the problem is considered at a high level of abstraction. Once the better solution for it is obtained at this level, a synthesis step occurs that converts the solution into a more detailed solution at a lower level of abstraction. Analysis of this new more detailed version is required to prove that it is an instantiation of the higher level of abstraction. Optimization again allows this version to be improved within the constraints of the higher level of abstraction.

An example of using different levels of abstraction is the way chemical processes are currently designed in industry. First very approximate models are used to select the general outline of the process. They tend to be task oriented. The process needs a task to purify the feeds to the reactor, the reaction task and then a task to purify the products and recycle the unused reactants. These task are later expanded into actual unit operations to accomplish them. On completion of a process design at the level of unit operations, one has developed what is termed the process flow diagram (PFD) for the process. It corresponds to a level of detail that can be drawn on one or two large sheets of paper. These sheets are then expanded into the piping and instrumentation diagrams (P&IDs) which show every piece of equipment which will be required to build the process. Here the

level of detail requires 50 to 60 large sheets of paper.

Each level of abstraction will usually present the solution in considerably more detail than the one above it. The more detailed level, however, loses the more global look at the "why" of the parts at which one is looking. Looking at the bits and pieces which make up the materials list for a distillation column, one may not readily understand it is a distillation column. Thus the higher levels of abstraction *are* often *needed* to explain the lower levels. Additionally, the reason a higher level decision is rejected is often based *on* constraints from the more detailed level, requiring the more detailed level to explain the *more* abstract level.

Sometimes abstraction is needed to understand a more detailed solution. An example is to "map" a multicomponent distillation column into a less detailed pseudo-binary (two component) column. Engineers can "understand" many aspects of a distillation column by using a McCabe Thiele representation *-for* it, but this diagram is only valid for a binary column. Here one is abstracting the behavior for a unit that is already supposed to be in the solution - i.e., one is going from the less abstract to the more abstract specifically to gain insight.

Another method to decompose a problem is to solve it at a fixed level of functional abstraction but by using simple models at first until one is close to the desired solution and then using more complete models to get to the final solution. This approach is a form of abstraction too, where the simpler models abstract the gross behavior of the more detailed ones.

An example is to solve a process optimization problem first using a mixed integer linear progamming model that suppresses most of the details and only approximates the behavior but still has the structure of the final process. This solution provides a first guess at the solution when solving again using more *detailed* models which can worry about rigorous physical property estimation and so forth.

This approach of using linear models at first has an *added* bonus of providing us with admittedly coarse models but ones that do not have local optima which *could* trap our optimization efforts.

EXPERT SYSTEM CONCEPTS

In this section we shall investigate the concepts underlying expert systems. We will answer the following questions.

- How *are* expert systems related to traditional programs?

- Why can computers do math? Can they also reason?

- What does it mean to say that expert systems ^*re* "knowledge based?"

- How do expert systems work, introducing the notion of models as symbolic structures and the mechanisms/strategies for manipulation of these structures.

Finally we shall deal with control mechanisms , representations and explanation -facilities for expert systems.

WHAT ARE EXPERT SYSTEMS?

Relations to Traditional Computer Programs

Expert systems *3ire* computer programs. We examine first how they differ from programs typically written in FORTRAN with which we *Are* more familiar: equation solvers, flowsheeting programs.

We all remember the two different types of mathematical homework problems we used to be assigned in elementary school. The first was to compute, for example, the value of 15*(37 - 14)/ (9 - 3*2). The second was a textual problem like the following.

> Peter has two jobs, washing dishes after dinner and keeping his room tidy. Each day he washes dishes he get 25 cents. Each *d3Ly* he keeps his room tidy, he get 1® cents, but each day he forgets he loses 5/ cents. How much money does he have after three weeks if he does the dishes every third day but forgets to tidy his room every second day?

We all remember that this second type of problem was much more difficult for us. Why?

In the textual problem we had first to figure how to restate the problem as a set of calculations that we could then solve using arithmetic. Then we had to perform the arithmetic. In some sense we can say that expert systems compared to traditional engineering programs *Are* like solving the textual problem is to solving the pure arithmetic problems.

While traditional programs perform calculations on structured well behaved problems, expert systems must in addition have the capability of understanding ill-defined, unstructured problems

and to transform them into more tractable problems, solvable by standard means, and then finally to solve them. This obviously requires the capability of reasoning.

We know computers can perform math, but can they be programmed to reason?

A machine by its nature can only perform in a mechanical manner. Tell it exactly what to do and it will do it. How can this capability be turned into one to perform math? The secret is in representation. The computer does not "know" that it is doing math. It simply is obeying instructions. When it encounters a "+" in a string of symbols, it has attached to the "+" symbol a set of instructions telling it exactly how to sum up the term on the left side of the "+" symbol with that on the right. In other words it is operating mechanically on a set of symbols.

If we look at formal logic, we see that reasoning can be very similar to arithmetic in nature. Examine the following two rules of logic.

1) $\forall x\ [W(x)] \wedge A \Rightarrow W(A)$

2) $W(A) \rightarrow W(B) \wedge W(A) \Rightarrow W(B)$

Rule (1) means that if W(x) is true for all x and A exists, then W(A) is also true. Rule (2) means that if W(A) leads to W(B) is true and W(A) is true, then W(B) is true. These rules are known severally as "universal specialization" and "modus ponens," and are two operators in formal logic.

There is really no difference here between logic and arithmetic. When the symbol --> is encountered in a string of symbols, and the computer recognizes that the left hand side of the --> is assigned the value of true, it can mechanically apply "modus ponens," carrying out the instructions telling it exactly how to assign the value 'true' to the right hand side of the --> . As an example of such a use, we could have the statement

Toxic(cyanide) --> Do not ingest (cyanide)

If the statement 'Toxic(cyanide) is true, the computer can assert the statement 'Do not ingest(cynanide)' must also be true.

This looks a lot like reasoning.

We are not intending here to infer that formal logic is equivalent to reasoning because that is probably just not true. However, it demonstrates that the concept of symbolic structures and operators to manipulate them have applications way beyond solving arithmetic problems.

The power of the computer does not lie in its complex hardware architecture, but in our ability to tell it how to respond to symbolic constructs. This "telling" can be done *in* many ways. When FORTRAN was introduced in the late 1950's, it broke the earlier monopoly of machine language programmers by providing language constructs understandable to people without any knowledge about hardware aspects of the machines which they were programming. In fact the first FORTRAN compilers were termed "automatic programming systems."

Today a wide variety of programming languages exist among which one *CAH* choose to solve a particular application. The choice reflects ones experience and belief that one is likely to be more useful than another for the problem at hand.

EXPERT SYSTEMS ARE KNOWLEDGE BASED

To call a program an expert system, it is not sufficient that it be able to reason. As the name indicates, fairly high performance standards *B.re* imposed on expert systems. They should be able to reason reliably and efficiently within a domain the way experts do.

Experts have a large body of information about their domain, and the[1]/ know how to apply it. Computer codes having encoded in them only factual information about a domain will not have the ability to distinguish between nonsense and fruitful application of this information. They will have to resort to search. The following example illustrates rather well the difference.

A five component liquid mixture of components A, B, C, D and E (A being the most volatile, B the next most and so forth) is to be separated into essentially pure component products using distillation. All components have reasonable vapor pressures at room temperature. The initial process to be considered will use simple two product only columns (no side streams) and each component will exit in only one stream (sharp splitting will be used). We wish first to find the set of columns which can accomplish this separation for the least utility cost; the columns can be heat integrated by exchanging heat among condensers and reboilers in the solution. This solution will give us a target against which to measure other solutions.

The A from B split is very easy, the B from C and D from E splits *Are* moderately difficult and the C from D split is very difficult but possible. Further the amount of A is small, the amounts of B, C and E *Are* moderate and the amount of D is large.