A retrospective on design and process synthesis

Arthur W. Westerberg*

Department of Chemical Engineering, Institute for Complex Engineered Systems (ICES), Carnegie Mellon University, Pittsburgh, PA 15213, USA

Received 24 September 2002; accepted 8 September 2003

Abstract

We discuss the impact over the past 40 years of process systems thinking on the design of chemical processes. We first explore the rich set of issues related to process design, only some of which are technical. We then briefly examine simulation, optimization and more extensively process synthesis ideas as they relate to design. Throughout we note that this progress is inextricably linked with the development of computer technology.

Keywords: Design; Process; Synthesis

1. Introduction

Since the mid-1960s, we have witnessed a rapidly increasing impact of systems engineering concepts on process engineering design that are largely coincident with the improvements in computing. We examine this impact, one that has very often come from the connecting together of well-understood parts to form a larger system.

The role of process systems engineering (PSE) is complex, ranging from new, powerful design methods, to how we think about the design process itself, to using operation issues in design decision making, to using design simulation methods to aid operations. As noted in an AIChE presentation by Perkins (1998), the entire issue of the AIChE Symposium Series No. 46 from 1962 usurped the title of PSE, and, in the introduction to that issue, we see the concept defined exactly as we are doing here. The term gained prominence in recent times as the name of a conference that took place in Kyoto, Japan, in 1982. No doubt we shall find even earlier references as the concept of unit operations, which apparently first appeared about 1915, leads to worrying about how to connect them together to form a process.

2. Designing

We start by asking what is the act of designing. Designing is often largely an art form. It is frequently the knowing of previous solutions, combined with guessing and then computing and/or proving experimentally that some new equipment arrangements might work better. This evolutionary approach can lead to some very sophisticated designs. However, with this approach, one can never be sure that much better designs are not possible. One goal of PSE is to provide methods that allow us to invent and then search among many more of the possible options.

2.1. Mathematical programming model

In his seminal AIChE Plenary Lecture and subsequent CEP article, Sargent (1967) proposed that we can look at design and operation problems as mathematical programming problems, albeit very large and difficult ones. And, as if to provide proof, we have seen since that time 10,000-fold improvements/decade in computer-based computational tools for solving such problems. We can now carry out industrially significant designs by searching for the best substructure embedded within a superstructure of allowed alternatives.

* Tel.: +1-412-268-2344; fax: +1-412-268-7139.
E-mail address: a.westerberg@cmu.edu (A.W. Westerberg).

© 2003 Elsevier Ltd. All rights reserved.

Keywords: Design; Process; Synthesis
This model, as powerful as it is, is of course not a complete model. We can only formulate design problems as mathematical programming problems only when we can a priori enumerate the design space of alternatives and only when we can connect cause with effect through a model.

2.2. Innovation

There is another aspect to systems design that involves innovation. What sparked the remarkable new design for manufacturing methyl acetate that Eastman Chemicals built in the mid-1980s (Agreda & Partin, 1983; Agreda, Partin, & Heise, 1990; Sirola, 1995, 1996). We must have another part of the design model, which supports creativity wherein very different, non-routine solutions appear. We need here, at the very least, the means to discover ways to interconnect existing technology to find arrangements that are not currently in our thinking. We have seen great strides in aiding innovation that have come from a better understanding of how to heat integrate processes, how to design separation systems and how to look at the design of reactor systems. The algorithms resulting are extremely useful when available as canned tools. But, even more important are the representations on which they are based. These representations allow designers who understand them to “see” the properties of the better designs. The designers benefit because they can directly construct such designs and/or because they can sharply reduce the design space they must then search. An anecdote may serve to make the point as to how much things have changed. In the late-1970s, this author visited a design team in industry and informed them of the ideas of Hohmann (1971) and Linnhoff and Flower (1978a,b), the latter just then appearing in the literature, on how to synthesize heat exchanger networks systematically. The next day, this team told my host that he had invited a “madman” to visit with them (albeit a quickly vindicated madman).

2.3. Not knowing the design space

But there is still more to systems design. We simply do not know enough about some design problems to allow us the luxury of knowing its space of allowed alternatives. We are not even sure of its goals, nor do we have any idea how changing the goals might affect the design solution found. For example, design a software system to support collaboration or design the design department in a multinational company. Or it is about 1942, and we have to think how to recover enriched uranium 235. Or it is 1970, and we want a plane that radar cannot see. We are asking how to attack design problems that no one involved in the project, and perhaps no one in the world, has previously faced.

In computer science, Simon (1981) classified these design problems as being ill-posed if we know the design space but not our goals and wicked if we do not know anything about the design space. We first have to convert them into problems we have a chance to solve. We have to establish a set of goals. We then have to discover even one, but hopefully many, design alternative that would be the start for the space of alternatives we wish to consider. We need to discover how to postulate a design and test if it is meeting our goals. Finally, we will be derelict if we do not discover what exists that is relevant. Biegler, Grossmann, and Westerberg (1997, pp. 10–18) discuss doing these tasks and relate it to doing a process design.

2.4. Understanding the goals

Another aspect to design is to appreciate the true goals for design. All design problems have multiple goals: be profitable, be safe, be reliable, and so forth. We always have to trade off our goals against each other. In addition, if we fail to formulate our objectives properly, we will almost certainly get the wrong solutions. We may state a goal as a constraint—e.g., we must make at least a 15% return on investment. We could easily miss a nearby design for which the return is 20%, a design we would in principle have found had we stated our goal to be to maximize return.

As another example consider what it takes for a new design to challenge an existing process. A new design has to be better economically than the existing process by paying back both investment and operating costs while the existing process only has to cover its operating costs. Not understanding this about the economics of a process can cause one to waste time looking at new designs that are only a little better than existing ones in a saturated market. As another example of understanding our goals correctly, we typically choose batch processes because they are much more flexible. As such we can design them more quickly and compensate for design errors by modifying how we operate them. A batch process to produce a single chemical throughout the year is virtually never more economical than a dedicated continuous plant.

2.5. Failure of the design process

Finally, the design process itself can often fail. A company can have all the right technology and miss producing the better designs because of poor communication among design team members.

Acting as an historian, Bucciarelli (1994) observed and reported in detail on three product design processes, noting just how complicated they were. He argues that design is a social process. One can see two interpretations for this observation. First, the actual work process is social. It involves being in meetings where one may spend inordinate amounts of time negotiating terminology, the project goals, how to make decisions, and so forth, with others whose backgrounds are very different. A second way design is a social process is that what we design reflects our personal and collective values and knowledge bases.
So how should we design better design processes? As with any artifact, we cannot improve the process of designing if we have no data on how it functions and on how well it functions. Based on experience, many companies establish policies and procedures for many of their work processes. Many companies have used computer-based support systems such as Lotus Notes. Typical use of these systems is to define a workflow that the company expects the designers to follow. The workflow prescribes, among other things, the types and sequence of documents the design activity should produce and the order in which people should review and sign-off on each.

Now comes the dilemma. Bucciarelli and others before him make it clear that problem solving, of which design is an instance, is anything but an organized definable sequence of activities. Even individual problem solvers move back and forth rapidly and almost randomly from one part of the problem to another. It is, therefore, not difficult to arrive at the conclusion that design support systems cannot be fixed in structure. They must support as the norm the continual deviation from that which is being prescribed. Designers will always be inventing a new document type. They will also need to capture new kinds of information and tie it with unexpected relationships to existing information. If for example one creates a database with fixed schemata and disallows changes, designers will be forced to operate creatively around this restriction or may in fact not use the database.

We were asked to think about designing the design process with our colleagues at a Process Systems Engineering Conference in Trondheim (Westerberg, Subrahmanian, & Reich, 1997). In this paper, we discuss converting this design problem from being wicked to being well-posed. We discuss the characteristics we see as necessary for design support environments. We are concerned about international organizations with distributed design organization trying to function effectively together. We postulate some goals. We point out that there is much room for improvement as we cannot at present capture and reuse very much of the intellectual capital created during the execution of our work processes.

2.6. In summary

As we discuss here, not all the systems aspects of a design problem have to do with mathematical programming or abstract representations. We need to understand system concepts whenever we strive to integrate together parts that we understand individually. We include here determining how we integrate a process within a company and how it affects the company’s goals. We also include how we put together work processes and people to carry out the design activity. If it seems a stretch to include them, then we justify our including them because our real goal is to understand how to design better processes.

We shall now examine in more detail the more technical issues raised here.

3. Design as a mathematical programming problem

If we know how to enumerate all the alternatives that we are willing to consider for a design and if we can evaluate these alternatives using appropriate quantitative measures, then we can in principle set up our design problem as a mathematical programming problem. In the extreme we can simply enumerate all the alternatives, evaluate them, and select the best design. There are several issues to worry about with this approach. First, we really need to understand a problem well—it must be routine—if we are able to enumerate all its alternatives. Our ability to enumerate requires we have a way to represent the space of alternatives. Such a representation may be the one real contribution of a Ph.D. thesis—i.e., they are not that easy to discover.

Second, to paraphrase Knuth (2001), $10^9$ equals infinity. The space of design alternatives is often so large it makes $10^9$ look small. The only way we can search such a large space is if we can take significant advantage of all the special structure and properties it may have, such as being linear and/or being composed of a number of parts that are sparsely interconnected. We succeed if we can find ways to eliminate large numbers of alternatives with one analysis, a situation that can arise if we can discover ways to bound the objective function values for families of alternatives.

3.1. Analysis

Advances in many areas in the past four decades have allowed us to make considerable progress in setting up and solving design problems as mathematical programming problems. There continues to be progress in our ability to simulate. In the past four decades, we have debated whether we should base these systems on a modular or equation-based architecture. Considerable evidence now makes it clear the best systems have the advantages of both approaches. We have evolved from solving 19 equations in 19 unknowns and using about as many hours of computer time in the mid-1950s, to models involving 650 equations in the mid 1970s (Lin & Mah, 1978; Mah & Lin, 1978), to the solving of 100,000 equations in a minute of 500 MHz computer time today. Both hardware and software have improved at enormous rates over the decades. Our understanding of how to create and manage complexity in modeling improved with the advent of object oriented concepts. Object orientation also leads one to the possibility of automatic modeling (Sargent, Vasquez-Roman, & Perkins, 1994; Stephanopoulos, Henning, & Leone, 1990a,b), where one might select the transport processes, state the assumptions...
that underlie the model and then have the system write the equations to model it. User interfaces are much more user friendly, allowing engineers access to very sophisticated software. We continue to hear about CORBA, \(^3\) OLE, \(^4\) ActiveX, and, in Chemical Engineering, pX (Balwin, 1996), Open Cape and Global Cape Open. \(^5\) The national labs are cooperating on the Common Component Architecture Forum\(^6\) project to develop software components for which one can use a graphical interface to wire them together to form a complex new software system.

New approaches to solving very large systems of equations robustly require the interface to pass a large amount of information about the model (see Drud, 1997 for a listing of such information for CONOPT). Our own work (Allan, 1997) indicates that we need even more information than Drud lists to speed up compiling, for example. This new information is not in any of the standards for defining solver objects.

Finally, a very difficult issue is the reuse issue for software. How can one improve the likelihood that others can locate and reuse the objects one creates? Reuse can range from reusing the object as is to allowing a reasonable end user to modify the code defining the object for a new situation. Are these chemical engineering issues? Should we be worrying about them? These will affect chemical engineering modeling in major ways, and we are in a position to make progress on them. We cannot ignore them.

An important ingredient of commercial flowsheeting systems is their ability to carry out the laborious task of computing repeatedly the physical properties for vapor, liquid and solid mixtures of species. Two issues are immediately important. The first is to have reliable, accurate methods for estimating. The second is to structure these calculations to support flowsheeting systems.

Models really have two major components in chemical engineering design: the part that characterizes the equipment and the part that characterizes the material flowing in that equipment. Physical property estimation is crucial for successful analysis of processes. We do pretty well on estimating the properties for hydrocarbon mixtures. When we push much beyond such mixtures, accuracy of the estimates is still a hard question.

3.2. Optimization

Two papers in this issue discuss the progress here to date (Grossmann, 2002) and the likely future progress (Biegler, 2002) in mathematical programming. We shall only highlight a few key thoughts that pertain to design and synthesis.

We not only want to analyze our processes, we also wish to choose equipment sizes and configurations to improve their performance. We have had considerable progress in mathematical programming over the past four decades also. Driven by the oil industry to schedule refinery operation, we first saw the development of large-scale linear programming packages in the late-1950s (where large is a relative term). First attempts at optimizing the design and operation of flowsheets were slow, requiring the equivalent of thousands of flowsheet simulations. Today, sequential quadratic programming approaches allow us to optimize the continuous decisions, often in the time equivalent to 2–3 flowsheet simulations. Progress in handling discrete decisions through the use of binary and integer variables has also progressed substantially. At first 10 binary variables was a large problem.

Today, taking advantage (often automatically) of special structure, we can often accommodate thousands of binary variables while using rigorous solving methods. When no rigorous approach seems to be available, many articles apply stochastic optimization approaches, such as simulated annealing and genetic algorithms to find solutions.

Envisioning that in about a decade we will be able to compute in a minute what now takes a week on massive parallel arrays of desktop computers, we should perhaps be posing a new question: how can we effectively use unlimited computing power to solve very large complex numerically stubborn design problems? One approach gaining considerable interest recently is to use cooperating systems of autonomous software agents operating in parallel (Siirola, Haan, & Westerberg, 2002; Talukdar & de Souza, 1994; Tyner & Westerberg, 2001a,b).

3.3. Heuristic approaches

When the problem is too large and we must solve it, we must resort to the use of ad hoc methods. Often we base these methods on observing the behavior when solving many problems of a given type. We might observe that the operating costs for a unit are negligible for a class of problems. Thus, we might solve future problems without considering operating costs. In distillation of nearly ideal mixtures, we may note that the better separation sequences always involve the early removal of the most plentiful species. We may, therefore, use rules to analyze a problem and to fix many of the discrete variables a priori to reduce the size of the search space. The ideal distillation literature started with the posing and testing of heuristics. Interestingly many of them do work well.

Masso and Rudd (1969) were among the first to propose finding heuristics through automatic learning when synthesizing heat exchanger networks. While it proved elusive for that particular problem, it was a very interesting idea. The learning literature is extensive in computer science. We have found the ideas behind the SOAR system (Newell, 1990) to be very interesting. All learning parameterizes a model from existing data. In our learning, we may have to pose and test

---

\(^3\) http://www.omg.org/.


\(^5\) http://www.global-cape-open.org/.

\(^6\) http://www.cca-forum.org/.
alternative structures for the model itself, which can give
the impression of our discovering things about the prob-
lem. Learning also involves guessing how to “variable-ize”
what we learn; for example, we may receive as input a tem-
perature with a value of 350 K and solve a problem using
it. Is the rule we are trying then to learn valid only for
that particular temperature or is it valid for any tempera-
ture? Finally, we may make mistakes in learning. To over-
come these mistakes requires that we occasionally challenge
rules we think we have learned so we can either remove or
override them.

We need heuristics to solve many of the problems industry
poses. Without them we need to formulate problems that are
too difficult to converge and/or too large to search. However,
we cannot guarantee the optimality of the resulting solution
when we use heuristics.

Rather than denigrating heuristics, we should observe that
they are everywhere. They are in our nonlinear solvers to
help decide which approach to try to get a difficult prob-
lem to converge. Heuristics are in our normal everyday de-
cisions when selecting the technology we will use to solve
a problem. They are there when we choose to use a form of
simplified model to characterize a batch process when de-
veloping a scheduling model for it. In other words, they are
there to aid in choosing how we should make decisions as
well as to which decisions we should make.

3.4. Evaluation support tools

Evaluation of design alternatives is itself a very interest-
 ing and complex subject. There are numerous measures that
characterize a design: cost, safety, flexibility, controllability,
operability and so forth. Ideally, we would like to reduce
all such indices to a common basis such as cost, but gen-
erally we cannot. Thus, we are always faced with making
trading-offs among the various incompatible indices. In ad-
dition, we often have no good way to evaluate these indices.
How should we assess the safety of a proposed design, espe-
cially during its early stages? We might look for hazardous
chemicals or extreme operating conditions—again, is this
one or two indices?

For multiple objectives, optimization theory can only tell
us how to establish which alternatives are in the so-called
trade-off or Pareto set. Finding a solution that improves
any index for a Pareto point requires that one of the other
indices must become worse. The engineer has to make the
final selection among those designs in the Pareto set based
on personal values. We can find the Pareto set by solving a
series of optimization problems where we “extremize” one
of the objectives while setting a variety of different bounds
for the others. For example, we might minimize cost for
each of 10 problems having a flexibility index no smaller
than 1, 0.9, 0.8, . . . , 0.1, respectively. The unique solutions
we find are in the Pareto set. There are also heuristic-based
algorithms that try to find a large number of the Pareto points
simultaneously (Deb, 2001; Kalyannoy, 2001).

3.4.1. Cost

The evaluation we all know best is to assess the eco-
nomics of a process. To assess the economic worth, we
need to estimate costs. We have methods to estimate the
cost of the equipment based on correlations developed from
past purchases (for example, Guthrie, 1974). There are com-
puter programs that characterize equipment by the number
of hours needed to fabricate it by different types of skilled
personnel and by the materials used in that fabrication. Input
to these programs are labor rates and materials costs,
and they can be surprisingly accurate. There are indices to relate
the costs in one year to those in another year. We can also
estimate labor rates, land costs, etc., for a process. Estimates
for the cash expenditures for a process design range from
plus or minus 50% down to 5% depending on the effort we
are willing to expend.

3.4.2. Economics

Given cost estimates, how should we compare the eco-
nomic worth of two alternatives? How should we compare
100 alternatives? If these are major projects involving signif-
icant portions of the resources of a company, then we have
to see how projects fit together over time to assess their im-
 pact. We cannot reduce the economic worth of the project
to a single number. We need to know the cash flow for it as
it varies over time. We need to choose those projects that fit
together best to maximize their estimated present worth to
the company while not taxing its financial resources. How-
ever, we will have trouble searching over 100 projects this
way. There are numerous ways we can carry out a project
over time and thus how its cash will flow over time is not
readily fixed. Thus, the search becomes astronomical in size.
When faced with a large number of projects, we are reduced
to assessing if projects are unlikely candidates by develop-
ing a single number for them—such as breakeven time—and
eliminating those that fail to pass a hurdle rate. The point of
this brief discussion is that, even for economics—which we
know best how to handle, the merit function is not straight-
forward to create and use.

3.4.3. Other merit functions

Grossmann and his students (Grossmann & Floudas,
1987; Grossmann, Halemane, & Swaney, 1983) have devel-
oped flexibility indices and ways to solve design problems
to maximize them. Their work shows us how we must
formulate these problems. Still, many problems that in-
dustry poses will tax our ability to carry out the resulting
calculations.

To assess the safety of a process has been the subject
of considerable study (see, for example, Caceres & Henley,
1976; Powers & Lapp, 1976). One approach is to posit a
safety problem such as an explosion in the reactor and work
backward through a fault tree to find what combinations of
“basic” events might cause it. If available, one can then prop-
agate the probabilities of these basic events through the tree
to assess the likelihood we may have an explosion. Another
approach looks at failures singly and together to discover which dangerous events their occurrence could cause. A whole literature (see, for example, work by Kramer & Finch, 1989; Suewatanakul & Himmelblau, 1995; Venkatasubramanian & Dhurjati, 1987; Venkatasubramanian & Rich, 1988) now exists on the use of expert systems to determine the common cause when a process fault occurs. Some systems use only rules, while others examine cause and effect models that the user supplies to the system. There is a preliminary study to assess the toxicity of a process by Grossmann, Drabbant, and Jain (1982). Perkins and Wong (1985) and others have developed ways to assess the controllability of a process.

4. Synthesis—representations for innovation

Process synthesis is very much the fun part of engineering. It is where one invents the structure and operating levels for a new chemical manufacturing process. One of the first publications on synthesis significantly predated the use of the word (Lockhart, 1947). It asked whether one should separate a ternary mixture by first separating the least volatile component. Rudd and co-workers introduced the term “synthesis” in the late-1960s (Masso & Rudd, 1969). The first review on synthesis referenced 66 papers and was by Hendry, Rudd, and Seader (1973). By the beginning of the next decade, Westerberg (1980) and Nishida et al. (1982) wrote review articles that referenced over 130 and 190 papers, respectively. A computer-based search of chemical abstracts for the terms “chemical engineering process synthesis” finds 466 relatively recent contributions (most of which are those intended).

The majority of the synthesis literature assumes we are constructing our processes using well-defined unit operations and that the problem is one of finding the best configuration among an astronomically large number of possible alternative ways to interconnect them. Major contributions have given us representations and algorithms to synthesize homogeneous systems, such as systems built of heat exchangers (Hohmann, 1971; Linnhoff & Flower, 1978a,b), of distillation columns (Hendry & Hughes, 1972; Thompson & King, 1972) and of reactors (Achenie & Biegler, 1986; Balakrishna & Biegler, 1992, 1996; Chitra & Govind, 1985a,b; Glasser, Hildebrandt, & Crowe, 1987). The early work of Suresh, Powers, and Rudd (1971) followed by that of Mahalec and Motard (1977a,b) showed us the potential of using artificial intelligence concepts to configure entire flowsheets. We shall highlight the major contributions for each of these topics.

Lastly, we see a growing discussion about constructing our processes at a more fundamental level—that is, by thinking of our processes as combinations of transport processes. If this approach successfully develops, then it will lead to designing not only the processes but also the unit operations themselves that should form the basis of these processes. We shall discuss these ideas in more detail when we discuss the synthesis of reactive distillation processes.

4.1. Different approaches

Biegler et al. (1997, pp. 33–35) enumerate the following approaches to synthesis: total enumeration of all the alternatives in an explicit space, a coordinated search in the space of design decisions, evolutionary methods, superstructure optimization, targeting, problem abstraction, and combinations of these. We have already mentioned some of these above when we discussed advances in mathematical programming, especially in the handling of binary variables, which correspond to the discrete decisions that allow us to select which types of equipment to use and to select how to configure that equipment.

4.2. Generating alternatives

The first real issue in synthesis is how to represent the design space of alternatives. The wrong representation can make the problem impossible to formulate and/or solve. The right may make formulation straightforward and may allow one to see how to solve efficiently. (Try enumerating all the alternatives for the exchanger networks one could propose to exchange heat between three hot and three cold streams where each stream can exchange at most one time with any other stream. You should get hundreds of alternatives. See Biegler et al. (1997, pp. 32–33) for a matrix representation that allows this enumeration.) The enumeration can be explicit, as with those described above or it may be implicit in the form of an algorithm to generate the alternatives.

Formulating a superstructure within which are all the alternatives one wishes to allow for a class of problems is one approach to use. Designing the form of the superstructure is often a major contribution. One does not want the superstructure to have the same substructure embedded in it in multiple ways as then one has guaranteed the optimization problem has multiple local optima. Agrawal (1996) shows that a very subtle difference in the proposed superstructure for heat integrated distillation increases the design space in interesting and important ways over the superstructure originally proposed by Sargent and Gamianbhandara (1976). The message: It is crucial to get the representation right. The right representation can enhance insights. It can aid innovation.

4.3. Search

The second issue in synthesis is how to search among the enormous number of alternatives, as we have mentioned earlier. Often search is the solving of a mixed (also contains real variables) integer (some of the variables to be solved for are integer) nonlinear program (MINLP). Using targeting as in
heat exchanger network synthesis or reactor synthesis, one may impose some very strong constraints that will eliminate major portions of the design space quickly. Using abstraction, one may make high level decisions that remove major portions of a space. An example is to decide to separate the alkenes from the alkanes first. Abstraction almost always dramatically reduces the size of the search space, with the drawback that one may throw out the best solution. To use it successfully one must have correct intuition about which decisions are the important ones to make first. As we shall discuss shortly, this type of decision ordering was a part of decision making in the AIDES system (Sirola et al., 1971), and it is central to the Douglas (1988) hierarchical approach to design.

4.4. Advances in synthesis

4.4.1. Heat exchanger networks synthesis (HENS)

The problem is to invent a heat exchanger network that allows one to recover and reuse process heat. The base problem is to invent a heat exchanger network having the least annualized cost that can exchange heat among \( n \) cold streams and \( m \) hot streams plus utilities. For this base problem know the flowrates and inlet and outlet temperatures for all process streams. Hofmann (1971) and Linnhoff and Flower (1976a,b) provided the remarkable observation that one could compute the minimum utility requirement for this problem without inventing the network. Subsequent work showed how to target the likely number of exchanges and even the capital investment required, again without designing the solution to the problem. These targets dramatically reduce the space of alternatives over which one needs to search.

Independently discovered by Umeda, Harada, and Shiroko (1979) and Linnhoff and coworkers (Linnhoff et al., 1982; Townsend & Linnhoff, 1983), the grand composite curve is one of the most insightful representations in all of process synthesis. This curve directly indicates the hottest temperature versus heat diagram we use for heat exchanger network synthesis, giving us a tool to visualize the alternatives (Andreovich & Westerberg, 1985). We can also represent the use of side strippers and enrichers and intercoolers and heaters on this same diagram (Biegler et al., 1997). Blending this representation with that for the grand composite curve allowed Linnhoff, Dunford, & Smith (1983) to show where one should not attempt to integrate a column into a process.

Duran and Grossmann (1986) provide a mathematical programming formulation to allow one to embed the minimum utility cost heat exchange within a superstructure model for an entire process. This model allows one to select the other units and their configuration, subject to there being a heat exchanger network that will use the minimum utilities. One has to be careful in solving the resulting problem, as it will likely have local optima.

Much work followed these early insights—e.g., to account for materials of construction of the exchangers, to account for differing heat transfer coefficients, to account for other than pure counter-current exchangers, to recover heat in batch processes where heat generation and recovery are functions of time and to allow for flexible process operation. An important review article on heat exchanger network synthesis is by Gunderson and Naess (1988).

4.4.2. Separation trains

As we noted earlier, Lockhart (1947) was one of the first to publish a paper on synthesis. He asked if one should separate the more or the less volatile species first when sharply separating a three component mixture into three pure components. During the 1970s, much work appeared on developing insights into finding the better separation sequences for separating relatively ideal mixtures of \( n \) components, generally into \( n \) relative pure single component products. The first contributions were on how to represent the design space (as a tree of alternative decisions).

Since columns require high temperature heat in their reboilers and release low temperature heat from their condensers, the problem of finding the best heat integrated sequences was also popular. The heat cascade diagram represents a column as a box whose width is the heat degraded in the column and height the temperature difference between the reboiler and condenser. We can draw it on the same temperature versus heat diagram we use for heat exchanger network synthesis, giving us a tool to visualize the alternatives (Andreovich & Westerberg, 1985). We can also represent the use of side strippers and enrichers and intercoolers and heaters on this same diagram (Biegler et al., 1997). Blending this representation with that for the grand composite curve allowed Linnhoff, Dunford, & Smith (1983) to show where one should not attempt to integrate a column into a process.

The separation of azotropic homogeneous and heterogeneous mixtures is another area in which very significant contributions exist. Since the early-1960s, researchers in the former Soviet Union have led in making important fundamental contributions to this topic. Fortunately, many of their works are available as English translations. Poellmann and Blass (1994) and Petlyuk (1998) highlight many of these contributions. The Russian work emphasized analysis and synthesis techniques based on looking at the separation trajectories in composition space. The literature in the west was spearheaded by Doherty and Perkins (1978, 1979) and then by Doherty and his students. We too (Wahnschafft, LeRudulier, & Westerberg, 1993; Wahnschafft & Westerberg, 1993) have worked in this area, specifically on synthesis methods. We now can teach methods to students to design distillation-based separation processes for ternary mixtures involving components that display azotropic and heterogeneous behavior. The key is again the representation, which, as just noted, is based on developing trajectories on a triangular composition diagram. The residue diagram shows the liquid composition trajectory that will occur in a packed distillation column while a distillation diagram shows the composition points that occur in a plate column. These diagrams show that so-called residue or distillation boundaries divide the space, and experience demonstrates that it.
Doherty, 1994, 1995; Ung & Doherty, 1995) showed that when destroyed, one could get past them in a single column. This could transform reactive distillation problems into a space and his students (Barbosa & Doherty, 1988a,b; Buzad & Doherty, 1994, 1995; Ung & Doherty, 1995) showed that one could transform reactive distillation problems into a space that made them look like a normal distillation problem. This work argued that reaction could destroy azeotropes, and, when destroyed, one could get past them in a single column. Hauan and Lien (1998) realized that one could study reaction, separation and mixing as additive vectors in composition space. The directions for the vectors are properties of the species only (and for mixing, the composition of the mixture) while the lengths are based on the design of the species only (and for mixing, the composition of the mixture). They wondered if one could compute all possible compositions one could reach given a set of reactions and their rate equations. This is like asking to find all the squares on a chess board that one can reach using a knight using only one step, then using two steps, and then three steps, etc. Reachability as an approach dates back to work by Horn (1964). Glasser et al. (1987) asked a different question. They wondered if one could compute all possible compositions one could reach given a set of reactions and their rate equations. This is like asking to find all the squares on a chess board that one can reach using a knight using only one step, then using two steps, and then three steps, etc. Reachability as an approach dates back to work by Horn (1964). Glasser et al. discovered one could derive properties of the reachable region. It is connected; indeed, it is convex. A point at the extreme cannot have a rate vector that points outside the region. There can be no rate vector outside the attainable region that points back into it. Originally, this work only considered isothermal processes, but they have shown that it is straightforward to extend it to non-isothermal conditions. Also they are able to argue that one can reach the extreme points of the attainable regions using combinations of the differential side stream, stirred tank and plug flow reactors only.

In a recent excellent article, Feinberg (2002) reviews the theoretical results available for attainable regions for reactor/mixer and reactor/mixer/sePARATOR systems.

4.4.4. Complete flowsheets

The goal for all of process synthesis is to discover the best complete flowsheet to accomplish a chemical manufacturing goal. Interestingly, one of the very early synthesis problems was just this one. Siirola et al. (1971) proposed the AIDES system to synthesize complete flowsheets using, among other things, means/ends ideas from artificial intelligence. A very significant contribution of this work was to order the design decisions into a hierarchy. Their system expected as input the possible reaction paths it might use. Given a reaction path, they selected the separation tasks, then the heating and cooling tasks and finally the pressure changing tasks. Equipment design followed based on the tasks selected. Mahalec and Motard (1977a,b) proposed their Baltazar system, where they used the unification principle from propositional calculus to develop alternative reaction paths, followed by separation tasks and so forth. Their approach differed in that all operators (reaction, heating, pressure changing, etc.) could reappear at any time when developing a solution.
Douglas (1988) produced a design text in Chemical Engineering based on his hierarchical decomposition approach to process synthesis. The essence of his approach was to put the decisions into an order, much like the previous work. His decision ordering was in levels: (1) choosing between continuous and batch, (2) selecting the raw materials and products, (3) selecting the reactor based on reaction selectivities, (4) designing the vapor and liquid separation systems, and (5) designing the heat recovery system. He wrote only the constraints that each level imposed on the design and used those to eliminate regions of space that could not meet those constraints. For example, level 2 only imposes reaction stoichiometries and selectivities. Reaction paths selected had to produce products that are more valuable than the raw materials. Many of his decisions were in the form of rules so the decision making could be quite quick. However, being rule based, it does not assure getting the best decisions.

Daichendt and Grossmann (1998) developed optimization approaches—based on developing bounds—to make the decisions for the hierarchical decomposition method of Douglas. One overriding message in all of the above methods is that the problem became manageable only when one imposed an order on the decisions to make. That ordering is largely based on making decisions first at abstract levels—eliminating major parts of the design space—and then at more and more refined levels.

In each of the above, there exists a computer program to illustrate the ideas. If one can convert the decision making into a program, then one has a routine design problem because the program must explicitly or implicitly state the design space. These systems can be very powerful and can lead to automatic design for well-understood problems.

4.5. Transport/thermo view

We have based chemical engineering design and operation largely on thinking of processes as interconnected unit operations. Unit operations first appeared as a concept about 1915. The methyl acetate manufacture process developed by Eastman Chemicals seems to be outside this way of thinking. It broke with tradition by not thinking about design as the configuration of known unit operations but rather as a set of tasks that one needed to accomplish in equipment yet to be selected, and it led to better than 80% reductions in costs because the program must explicitly or implicitly state the design space. These systems can be very powerful and can lead to automatic design for well-understood problems.

Our hypothesis then is that we might find future design methods based on thinking in transport space that will give us significant process improvements. Rather than using existing unit operations, we will be indirectly inventing new ones tailored for the design at hand. Perhaps, with time, we will simply settle on a new set of more versatile unit operations and return to the unit operation approach for design with much improved results.

4.6. Retrofit

The last topic we discuss on design itself is that of retrofit versus grassroots design. We have argued (Grossmann, Biegler, & Westerberg, 1987) that retrofit design must be more complicated than grassroots design as the retrofit design space includes doing a grassroots design as an alternative. Actually, the retrofit problem is in principle combinatorially much, much larger than a grassroots design. In retrofit design we not only enumerate the design alternatives, but then, for each alternative, we have to decide which of the existing equipment we can reuse and where. Of course, we can tame the problem by limiting how much of a process we are willing to redesign or by limiting the budget to be a small fraction of the cost of a new plant. Many capital intensive projects in the last two and half decades have been retrofit designs. The research community should spend more time investigating how to approach these problems. We can and should argue, of course, that what we learn to aid grassroots design also gives insights needed for retrofit design. However, the assigning of tasks to existing equipment leads to a different type of analysis. We need to know if the equipment can function successfully when it is markedly away from its normal operating range. For example, will the assigned column weep when it is much taller and wider than we need? We should note that flexible operation of new designs also leads to these same modeling issues, so we need some common capabilities in modeling for both.

5. In closing

We have looked at the impact of PSE on the design of chemical processes. The impact became possible with the advent of computing. Desktop computing has attained “giga” rates for both instructions executed per second and bit transfers over our inter- and intranets and giga sizes for fast memory. In June of 2000, IBM announced a 12.3 teraflop (10^{15}) computer, over 1000 times faster than the fastest PCs at that time. What we will be able to do (beside playing chess) with this power is still largely unexplored and likely not yet even in our thoughts. One thought is that we are moving...
from computers being super computers to computers being information processors.

We tried to show that design is multi-faceted. Some designs are routine, and one can fully automate the process to perform them. However, for others we are not able to conceive of the design space and thus must support innovation by designers. When we can propose the design space, we can often formulate our problem in mathematical programming terms. However, even for these problems, we struggle with formulating valid merit functions with which to compare designs and therefore we spend a great deal of time solving, picking the better ones from the results, rather than limiting ourselves to trying only one.

The views throughout this paper are limited and heavily biased with the author’s understanding of these issues. As with any overview, there are inevitable and almost certainly key omissions and misunderstandings, and, for these, sincere apologies are extended.

References


Westerberg, A. W., Subrahmanian, E., Reich, Y., & Konda, S., the n-dim group (1997). Designing the design process. Computers in Chemical Engineering, 21(Suppl.), S1–S9.