A KNOWLEDGE BASED SYSTEM FOR THE DESIGN OF RO-DESALINATION PLANTS

K. PAPAFOTIOU, D. ASSIMACOPOULOS and D. MARINOS-KOURIS
National Technical University of Athens, Dept. of Chem. Engineering
* Present Address, Intracoft S.A.

Summary

The prototype of a Knowledge Based Expert System for the design of RO-Desalination plants is presented. An Object Oriented Software development approach has been followed so that the system be open to further improvements. The behavioral part of knowledge is expressed in the form of qualitative IF-THEN rules. The design method is based on a class of objects which encapsulate the behavioral knowledge of the RO-Plant as well as the mechanisms for its exploitation.

A two-level hierarchical design procedure was adopted. On the first level the general characteristics of the plant are defined. On the second level, the plant structure is determined by first designing the high pressure stage. The design process continues adding more stages to the plant structure until the product quality conforms to the input specifications.

1. Introduction

In the present work, experience gained from the area of Computer Aided Process Design (CAPD) is applied to the field of Water Desalination. The synthesis of an RO-Desalination Plant, i.e., the automated invention of the plant structure and determination of the design variables is attempted. For this purpose, a Knowledge Based Expert System (KBES) is developed and tested.

Computer aided design attempts to solve problems which display the following characteristics, [1]:

- They are under-defined.
- They are hierarchical in nature.
- Solution(s) should be sought in a large search space.
- The overall solution emerges from the solution of subproblems.
- The artistic nature of design, results in the lack of strong standards in the methodologies which are followed during either the synthesis or the analysis phase.
Previous experience on the development of CAPD systems, suggests an open system architecture which permits the integration of several types of knowledge (heuristics, short-cut calculations, models of several levels of accuracy) and appropriate mechanisms for its exploitation.

A system able to comply with these specifications should look like a shell. Several types of knowledge related to the problem and the solution strategy would be integrated in this environment. Tools, such as steady state simulators, optimizers, graphical user interfaces, data-banks for physical properties and cost information, would support both the synthesis and the analysis of the plant.

The traditional top-down software development method begins with the gathering of user requirements and preparation of functional specifications for the application. In a second phase, iterated procedures are used until a convergence between the user specifications and product performance is achieved. This approach appears to be inadequate to face the additional requirements introduced by the complexity of an open architecture:

- on the design and implementation of the system and individual tools.
- on the methodology of system development.

The characteristics of the design phase make it very difficult to set specific requirements on the system and proceed to its development. A rapid prototyping approach seems to satisfy better our criteria of system development. In this approach, one gathers again the user requirements and constructs a working prototype according to them. This prototype is further refined in successive iterations between the developer and the user.

This last approach will be followed for the development of an RO-Desalination Plant design system. The following points will be discussed:

b. Knowledge required and its representation.
c. Methodology for software development and associated tools.
d. Finally, the system will be tested against the synthesis of a simple design structure.

2. System Requirements
2.1 Design methodology

A two-level hierarchical design methodology was adopted, having to solve a number of subproblems in each level.
Level 1: General characteristics of the plant.

- Input/Output characteristics.
- Overall conversion.
- Recycle requirements.

The information which should be provided to the system at this level, refers to the raw feed characteristics, product requirements (flowrates and fluctuations over a year, quality, etc.) and cost items.

Level 2: Generation of plant structure.

At this level, the design of the first (high pressure) stage is realized. The design process continues with the generation of more stages (product and/or brine stages) as long as product specifications are not met.

At the second level, the decisions for each stage include:

- Membrane and pressure selection.
- Stage conversion.
- Stage product stream destination.
- Stage brine stream destination.

2.2 Knowledge Requirements

The above described methodology requires the following types of knowledge to be already embedded in the system:

i) Knowledge concerning process objects (membranes, streams, stages, etc.) and concepts. As an example consider:

The object: membrane
The concept: membrane operates at high pressures

ii) Knowledge concerning the relationships between objects, i.e., the topology of the process.

iii) Taxonomies, that is, the general classes of objects. One possible classification of membranes, [2], is shown in Figure 1.

iv) Behavioral knowledge based on heuristics and models (either quantitative and/or qualitative). Behavioral knowledge can be expressed in the form of IF-THEN rules. As an example, consider the following:

\[
\text{IF pressure increases} \\
\text{THEN water flow through membrane increases}
\]

The three first categories are referred to as the "static" part of knowledge, while the last as "dynamic knowledge" in order to emphasize the fact that the latter creates objects and gives specific values to their attributes.
3. System Development
   Object Oriented Programming environments satisfy the requirements posed by the complexity of the open architecture design, [3]. An Object Oriented system has the following characteristics:
   - The entities occurring in the problem are represented by data structures called objects.
   - An object is an encapsulation of data (the entity properties) and procedures, usually called methods.
   - Methods are invoked by sending messages to objects.
   - Similar objects are grouped in classes.
   - Classes have a hierarchical organization.
   - Objects which are instances of lower classes inherit data and methods from their superclasses.
   - Instances of different classes have the ability to respond to the same message.

The object oriented approach allows for:
- The quick implementation of successive working prototypes in a consistent manner.
- The provision of closing the gap between the user's abstract idea about the application and the detailed view that the system developer has. By the term "abstract" we mean the identification and manipulation of the entities occurring in the problem according to their external behaviour.
- The use of several parts of the system, as reusable software components, in different applications and the easy integration with other tools.
- An easy extensibility and modifiability of the system.

The present work was implemented in the SMALLTALK/V programming environment, [4], and it was tested on an IBM/PC-AT computer.

3.1 Software Development - System Overview
Software engineers usually restrict their collaboration with the domain experts to the analysis and knowledge acquisition phase. This practice often leads to a different treatment of the physical problem from the actual software implementation. The formal deduction mechanisms of the software implementation phase have little notion about the specific physical domain and therefore should be completed with additional procedures for the verification of integrity and consistency of facts.

In order to overcome these problems, a unified procedure was followed and the results of the analysis were directly transferred in the design and implementation stage. This can be realized easily in the Object Oriented environment adopted, because of the ability to define classes of objects representing directly a designer. The internal structure of such an object (designer-object) contains all the information that makes someone expert in the field, that is:
- knowledge on the methodological level,
- knowledge on the physical level.

3.2 Knowledge Representation
3.2.1 Representation of Static Knowledge
In an Object Oriented environment knowledge representation means:

- Identification of object classes.
- Arrangement of these classes in a hierarchical manner.
- Establishment of object interrelations through the definition of their internal structure.
- Establishment of communication channels through the definition of message interfaces.
Identification of Objects

For the RO-plant the following basic object types have been identified:

i) Membranes
ii) Streams
iii) Stages
iv) The whole plant

Classification

The need for the representation of taxonomies is anticipated by organizing classes in a hierarchical manner. The complete hierarchy of classes used in the RO-plant design is shown in Figure 2.

![Figure 2. Hierarchy of Classes](image)

Internal structure

The four basic classes are related to each other by: "isPartOf:"

relations, according to the diagram of Figure 3.

Message Interfacing

One of the most essential characteristics of the Object Oriented
Systems is the encryption of the internal details of each object. Access to the object structure is achieved by the definition of a set of procedures, called methods. These methods make use of the properties encapsulated in the object structure and can be executed by passing their names as messages to the object.

The methods defined for each class (as in Figure 3):

- Handle relations between objects (such as those occurring in the diagram of Figure 3), e.g. assignment and return of values.
- Perform exception handling and error prevention (e.g. do not allow an improper kind of stream to enter a stage).
- Perform functions on objects (e.g. mass balances).

![Figure 3. Relations between Objects](image)

### 3.2.2 Representation of Dynamic (Behavioral) Knowledge

For the purposes of the first Working Prototype, behavioral knowledge is expressed only in the form of IF-THEN rules. The rules included in the Prototype represent knowledge coming from:

- Models expressed in qualitative form.
- Experience (heuristics).
- Constraints set by the available equipment, feed, cost, etc.

As an example of a rule embedded in the developed prototype,
consider the following:

Rule

number: #chrl
condition: #((CurrentStage conversion = 'medium')
& (CurrentStage inletStream flowRate = 'medium-high'))
action: #((CurrentStage brineStream flowRate: 'medium'))
description: 'Brine Stream Flowrate is Medium'

3.3 Representation of Design Methodology

The different design steps described in 2.1 are represented by the ordered collection of tasks encapsulated in the structure of the designer objects. These tasks are executed sequentially:

i) For the first level of design.
ii) For each stage during the plant structure generation level.

Each task is composed by a forward chaining inference mechanism and a set of suitable rules associated with it. When a task is invoked, the associated rules are activated and the corresponding subproblem is solved.

4. Test Study

The first level of the design procedure is mostly user interface driven, involves mainly inputs from the user and requires the presence of a database containing cost information. Here we shall be confined in the presentation of the second level decisions (generation of plant structure).

The information passed to this level is the following:

- The feed consists of brackish water.
- The flowrate of the feed is high.
- The overall conversion is high.

Under these specifications, the decisions for the first (high pressure) stage would be:

- A brackish water membrane operating in medium pressure should be used.
- A medium conversion should be achieved and thus, the flowrates of product and brine streams should be medium too.
- No further treatment needed for the product stream.
- A brine stage should be used to exploit the brine stream since it is still brackish.
At this point, the plant structure contains a new stage to be processed in subsequent steps (Figure 5). The decisions for this stage are:

- The same membrane as for the previous stage should be used (the pressure of the brine stream is almost the same as for the first stage).
- The conversion is medium.
- The flowrates of the product and brine stream are now medium-low.
- A brine stage should be used for further exploitation of brine stream which now is high-brackish.

Accordingly, a third (brine) stage is added to the plant (Figure 6). The decisions are similar to those of the previous stage except that the flowrates are low and the brine stream is now seawater and cannot be exploited any more. The design procedure stops at this point.

The overall performance of the plant is not checked because the rules are assumed to be constructed so that the characteristics of each individual stage are consistent with those of the overall plant.
5. Conclusions

Throughout the development phase of the prototype KBES described above, a real problem was used as a case study. Our intention was not to test the capability of the system to cope with difficult design problems, but to assure the adequacy of the adopted model.

The main achievement was the specification of an information structure which closely resembles to the concepts appearing in the physical prototype. This allowed the rapid development of prototype KBES which can easily be incorporated in a complete design system for RO-Desalination Plants.

Our current work is concentrated on two directions:

i) On the engineering level:
   Refinement of the design methodology and enrichment of the knowledge included in the system with both models and heuristics.

ii) On the software level:
   ii.1) Expansion of the Knowledge Based system to include more equipment object classes and to accommodate types of knowledge not included so far, such as quantitative models.
   ii.2) Refinement of the System Architecture in order to obtain an Open Nature where the core Knowledge Based System could be easily integrated with databases, analysis tools, user friendly interfaces, etc.

6. References