

Object Oriented Tool for Modelling, Simulation and Production Planning in Petrochemical Industries

Kary FRÄMLING*, Leif HAMMARSTRÖM**

(*) Ecole des Mines de Saint-Etienne

158, cours Fauriel, 42023 Saint-Etienne Cedex 2, FRANCE

Tel. 77.42.02.39, Fax. 77.42.00.00

E-mail: framling@emse.fr

(**) Neste Oy

P.O.B. 310, FIN-06101 Porvoo, FINLAND

Tel. 358-15-1873145, Fax. 358-15-1877221

E-mail: lgh@jukka.neste.fi

Abstract:

The subject of this work is to study the use of object oriented models of petrochemical plants in order to simplify simulation and production planning. A prototype system for an existing plant was developed to achieve this goal.

The prototype system was made using the object oriented programming language Smalltalk/V. The system is generic and allows creating graphical simulation models for most kinds of petrochemical plants mainly as a simple drawing operation. The models can be used for simulating production plans as well as for generating them automatically.

The results obtained prove that object oriented methods allow the creation of very flexible models. These models are shown to facilitate simulation and automatic scheduling, when used together with artificial intelligence methods.

Keywords: Object Oriented Models, Chemical Production Plants, Smalltalk, Simulation, Production planning

1. INTRODUCTION

Object oriented (OO) models for the simulation and production planning in the petrochemical industry are studied in this article. A prototype system of a general modelling tool was built, which was used for constructing different models of the test plant (a phenol production plant). The goal of the work has been to find solutions that would be as generally applicable as possible for different kinds of production facilities.

The work has been financed by Neste Oy (Finland). All data that might be considered confidential has been modified in this article. However, this does not change the contents of the article in any way.

1.1. Related research

The development of efficient models of production facilities has always been (and still is) limited by the computing resources available. As these computing resources become more powerful, it is also possible to introduce new modelling methods. One of these methods is the OO modelling and programming.

Production plants are very complex systems, which require management of huge amounts of data. The main advantage offered by OO models is that it is possible to localize this data into manageable, independent units. This means that the model itself already contains a certain amount of "intelligence", more than that found in classical models. OO programming has been used for describing chemical processes for instance by [Hammarström, 1990; Henning et al, 1988; Lakshmanan and Stephanopoulos, 1988; Stephanopoulos et al, 1987].

Production planning using OO programming is also becoming increasingly popular, partly thanks to the development of graphical user interfaces [Alasuvanto et al, 1988; Graves, 1981; Jensen, 1986; Stephanopoulos et al, 1987; Stephanopoulos, 1987, 1988]. Combining OO programming with Artificial Intelligence methods has turned out to be very useful for certain production planning problems [Falster, 1987; Realff, 1989, 1990].

2. OBJECT ORIENTED MODELS OF PETROCHEMICAL PLANTS

Different kinds of computer models are excellent tools for analyzing real world phenomena [Stephanopoulos, 1987]. However, when these real world systems grow complicated it often becomes very difficult to find all equations and relations between different variables of the system. Models are normally constructed for varying purposes, which require different data and calculations.

OO systems offer a natural solution to this problem of data and calculation management, since the data for each operation is collected to the place where it is needed (the object), together with the calculations/operations that can be performed with this data (the methods).

The three main properties of OO systems are inheritance, encapsulation and polymorphism. When modelling chemical plants, inheritance is useful for defining common properties of different kinds of equipment (heat exchangers, distillation columns, etc.). This reduces the programming effort required. Encapsulation means that the data of the model is completely associated with objects, which control the access to it, thereby preventing programming errors or corrupting data. The polymorphism means that it is possible to have generic messages that can be sent to every object of the model, which will handle them in their own way, therefore reducing the complexity of the program.

The programming tool used was the OO language Smalltalk/V for Apple Macintosh. Smalltalk is still the most object oriented language of all programming languages, so it offers all the advantages of this technology. The system had to be usable on hardware already existing at the plant (which excluded the Symbolics system available), and some portability was desired (Smalltalk/V also exists for DOS, Microsoft Windows and OS/2). The presence of a Prolog module (Prolog/V) was also considered, even though it was never used.

2.1. Object classes of a process model

The choice of object types and classes depends mainly on the purpose of the model. In the case of production planning, global factors such as production capacity, storage and transportation facilities are crucial. It is, however, also necessary to have a detailed model of the plant, especially in order to be able to simulate the effect of errors in different units on the production plan. Real time error detection would be useful for finding these errors early enough.

The following kinds of equipment can be modelled with the prototype system:

- Tanks
- Flows
- Harbours
- Road-vehicle terminals
- Groups of production units
- Production units

These classes are only meant for the needs of the production planning. Real time error detection would require more specialized classes.

2.2. Properties of different classes

The different kinds of units are defined as classes in Smalltalk/V. The class hierarchy is shown in figure 1.

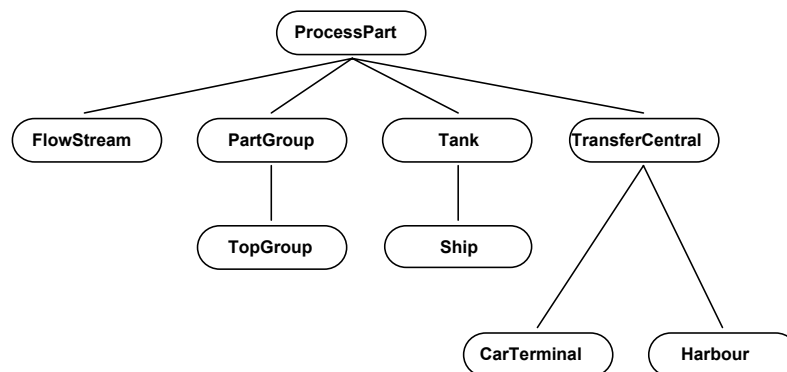


Figure 1. Class hierarchy for process parts.

Common properties of all parts - the class "ProcessPart": This class is very general, although it already contains everything needed for modelling production units. This is why no special class "Production unit" has needed to be defined. All the other classes are merely specialisations of this class. The class has the following main properties (examples of other properties are the name and icons):

- Efficiency factor, which describes the operational state of the unit.
- A lower limit for the efficiency factor, below which the unit becomes non-operational.
- The products (qualities) that can be produced by the unit.

- Incoming flows.
- Outgoing flows.
- A production plan containing product changes.
- Schedule for periods with lower efficiency (maintenance stops etc.).
- Logic variables indicating if the unit has private production plans and period schedules, or if it uses those of the partgroup to which it belongs.

Flows: The class "FlowStream" represents flows between different units. No capacity limits have been defined for flows, even though there are obviously some. A limit may, however, be introduced by adding a supplementary unit to the flow.

For the sake of simplicity all flows are directed, meaning that they have one input end and one output end. This is not necessarily true in a real chemical process, but one bi-directional flow may be represented by two directed ones. This limitation is quite common in this kind of model [Stephanopoulos, 1987; Hammarström, 1990].

Flows have the properties (in addition to those of "ProcessPart") content (product mixture), density of the liquid, pressure, temperature and viscosity. None of these properties has any importance for the production planning purposes, but have been included for completeness.

Unit groups: The class "PartGroup" groups together units into logical entities, which makes the building of hierarchical models possible. Connections to a unit group are done exactly as to an individual unit. These external connections are shown as special symbols within the unit group, to which the units of the group can be connected.

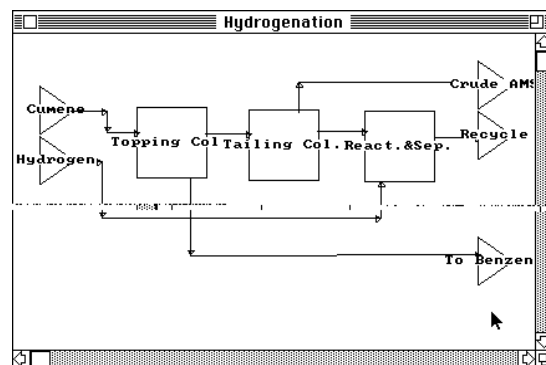


Figure 2. Unit group. External connections are shown as triangles.

The unit groups have the following main properties:

- Knowledge about to which model it belongs (i.e. which top group, see below).
- A collection of unit symbols.
- A display window.
- A background image for the window.

The highest level of the model: Every model contains one object of the class "TopGroup", which is the highest level in the unit group hierarchy. Its properties are the same as those of any unit group, with the only limitation that there can be only one per model. The main difference between a top group and any other unit group is that the top group is used as an entry point for having access to all the objects of the model.

Storage facilities: The class "Tank" represents a storage facility of some kind. In practice, all units of a chemical plant have a storage capacity (or delay), but this is not important for the use of the model.

The properties of the class are the following:

- Total volume.
- Minimal level for loading onto ships.
- The last measured level and the time of measurement.
- Internal variables.

Ships: Ships are essentially storage places, which allow products to "disappear" from the model. They also have a certain loading speed, which is important to take into consideration.

Transport terminals: Transport terminals belong to the class "TransferCentral". In addition to the properties of the class "ProcessPart" they also have a transport schedule.

Road-vehicle terminals: Road-vehicle terminals can be considered to have quite a stable throughput, with occasional peaks. Therefore they behave as production units, except when their transport schedule says otherwise.

Harbours: Products can pass through harbours only when a shipment is programmed into its shipment schedule. The shipments can be thought of as large "batches".

2.3. Creating a model

A new model is made by creating an instance of the class "TopGroup", which opens a "drawing window" where the model can be defined. The model is built by drawing icons representing units into the window. The fact that building the model is essentially a drawing task simplifies the creation phase significantly. The user never needs to see any program code or remember complicated commands.

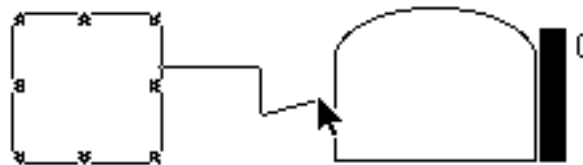


Figure 3. A tank, a production unit and a connection that is being created.

Units are created in the special editing mode, where they can be created, copied and destroyed. For adding a unit the user first selects the type of unit, which determines the unit's properties and icon. New units created through duplication have the same properties and icon as the original one. Even some property values are copied, which speeds up the creation of the model. Units are connected by drawing directed lines between them.

Hierarchical models, where the model gets more and more detailed on lower levels of the hierarchy, are useful because the same model can be used for several purposes. The model hierarchy is defined in a "top-down" fashion.

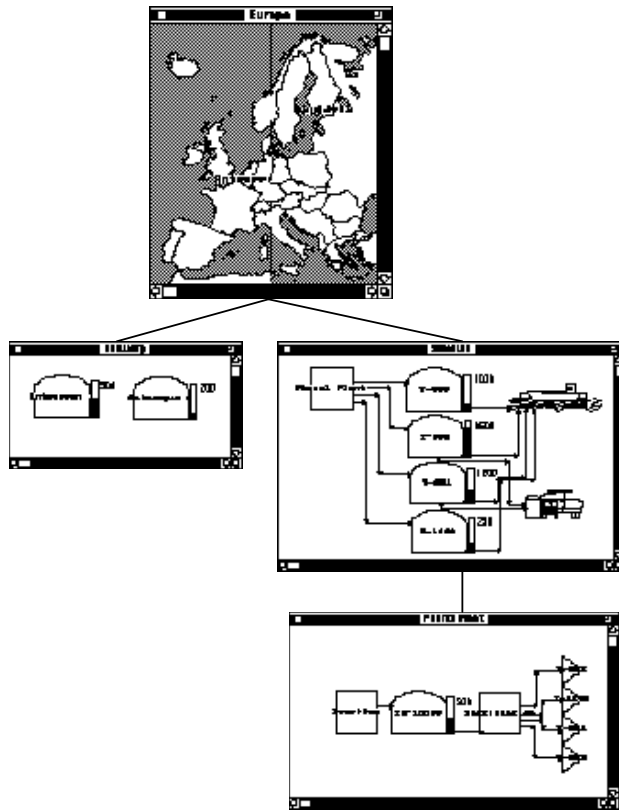


Figure 4. Hierarchical model of a phenol plant and storage places.

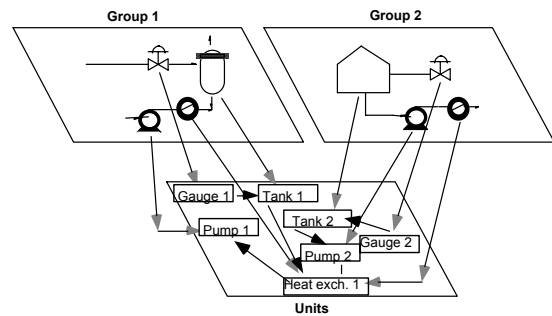


Figure 5. Relation between unit groups, icons, model and units.

The connections in chemical plants are often very complex, which means that the model rapidly becomes practically impossible to understand. If the same unit could be included in several unit groups, the number of unit groups would probably increase, but the complexity of each would decrease radically.

The problem has been solved by allowing several icons for each unit. This means that different icons can be placed into different unit groups, where only the connections to the other units of the same group need to be shown. The relation between icons and units is shown in figure 5. The two unit groups contain common units, but the icons map to only one unit in the real connection model.

2.4. Use of the model

Once the model has been built, the property values of all units have to be defined. A unit-dependent menu appears when selecting a unit, which shows all the actions possible.

Unit properties are defined through Macintosh dialogues. Since different unit types have different properties, they also have partially different property defining dialogues. An example of such properties are the ratio of the total flow for each input and output connection, which has to be defined in order to be able to calculate the product flow. Another property of the same type are the functions which indicate set-up times of the unit when changing from one product to another.

It is also possible to use the model for getting the product flow through each connection at each time. This means that it is possible to get the production capacity of each product at any time.

3. SIMULATION

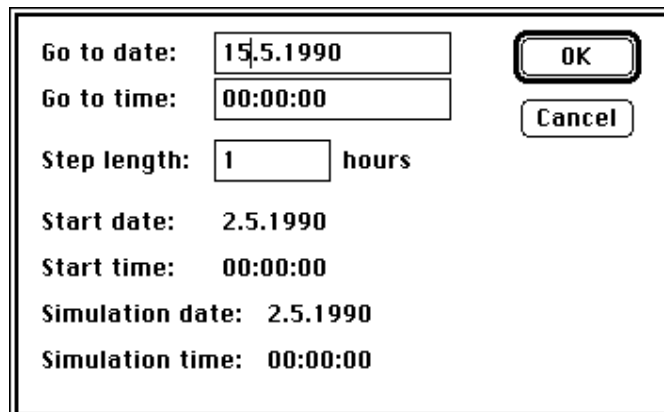
Simulation allows the user to test different production plans in order to find an optimal one. What is actually simulated are the tank levels, since these are the critical indicators of the success or failure of the production plan. The plan has failed if a tank either runs empty or becomes full.

3.1. Handling of real time aspects

Calculating the production capacity (or material flow in each pipe) is possible to do with a maximal precision of one hour. This calculation is, however, complicated by the presence of several feedback loops between units of the plant. The method used is presented in [Främling, 1990] and calculates the flow through all the connections of the plant. Since the production capacity of the plant can change at any moment due to a period of reduced capacity in some unit, any estimation of the production capacity for more than one hour can only be a guess.

However, the simulation does not have to be done at the same precision. When a tank is included in the simulation, it calculates its level at simulation step $t+1$ as (the level at time t) + (the incoming amount during the simulation step) - (the outgoing amount during the simulation step). No test is done if the tank actually runs empty during this time, so all tanks are supposed not to become full nor run empty during a simulation step. Therefore the simulation step length does not have any impact on the simulation precision, except if a tank actually runs empty or becomes full. A simulation step longer than one hour is useful as it speeds up the simulation significantly despite the fact that the production is still calculated with a one hour precision.

3.2. Launching a simulation



The image shows a simulation dialogue box with the following fields and buttons:

Go to date:	<input type="text" value="15.5.1990"/>	<input type="button" value="OK"/>
Go to time:	<input type="text" value="00:00:00"/>	
Step length:	<input type="text" value="1"/> hours	<input type="button" value="Cancel"/>
Start date:	2.5.1990	
Start time:	00:00:00	
Simulation date:	2.5.1990	
Simulation time:	00:00:00	

Figure 6. Simulation dialogue.

Before starting the simulation, the user selects the tanks that he wants to include in the simulation. The starting time of the simulation depends on which tanks are simulated, because it has to be started from the oldest real level indication. Tanks with a newer level indication are not included in the simulation until their real level indication time has been reached.

It is possible to change the simulation step length during the simulation, which means that it is possible to focus on critical periods. The simulation can also be advanced step by step and backed up. Backing up is especially useful, as it makes it possible to change the production plan, and test it again without having to restart the whole simulation.

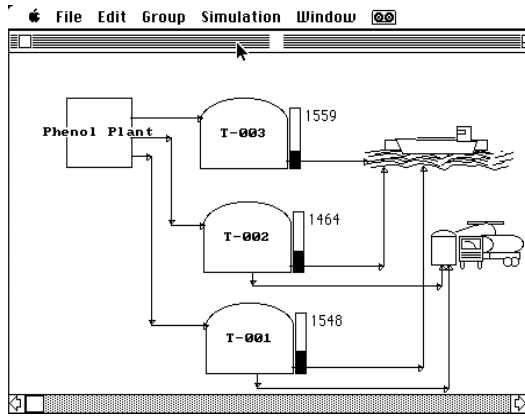


Figure 7. Ongoing simulation with continuous level indication.

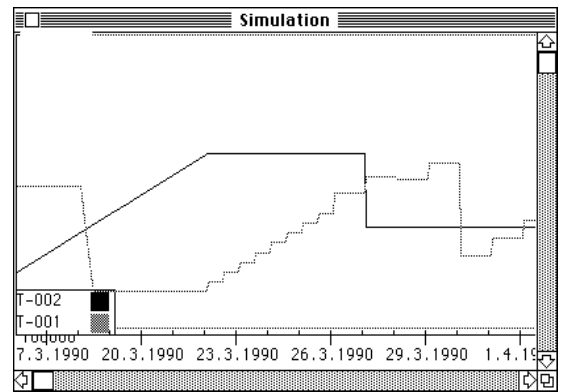


Figure 8. Simulation curves. The simulation step length has been changed twice (12h and 24h), which is why the curves are less smooth at the end.

The simulation results can also be studied as curves, which helps finding critical situations. The production and shipping plans used can be seen quite clearly from this curve. The tank's curve starts to go up during manufacture of the product that it contains. Sudden drops in the level indicate shipments. A slowly decreasing level indicates transport by road-vehicle.

4. AUTOMATIC PRODUCTION PLANNING WITH THE MODEL

The production plan is produced by combining the simulation with the planning module. The planning module consists of several product specific sets of heuristic rules. Six different margins are used for calculating the heuristic estimate for each product. The product with the highest heuristic estimate is the one produced.

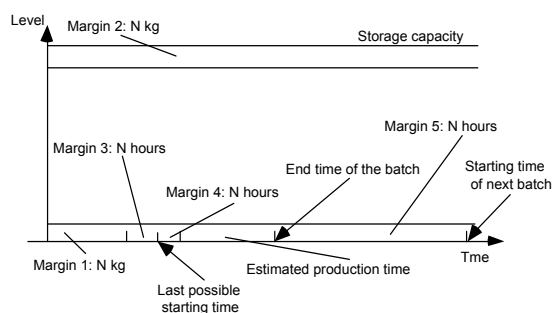


Figure 9. Security margins 1-5.

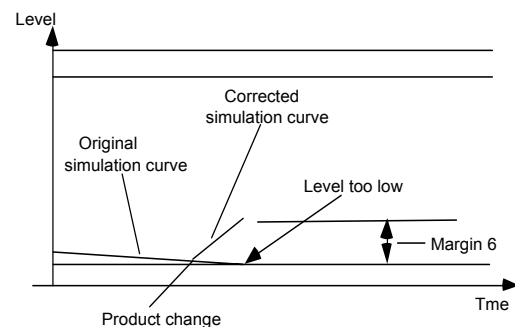


Figure 10. Margin 6.

The margins used are the following:

- Margin 1: The lowest level allowed in the tank.
- Margin 2: The safety margin for the highest level desired in the tank.
- Margin 3: The time before the last possible starting time when the production should start.
- Margin 4: The last possible starting time for producing the next batch, taken into consideration the product change delays.
- Margin 5: The time that the product should continue to be produced even when not needed for the next batch.
- Margin 6: Safety quantity that should be produced when the production plan fails and is corrected (mainly for failures due to road-vehicle transports).

The times required for producing the needed quantity (margin 4) are estimated by "asking" the model how long it thinks it will take to produce the quantity, production stops and disturbances taken into consideration. Margin 3 is added to this time estimate to be sure to start the production in time. Unfortunately it is not possible to give an exact estimate of the production time due to the great amount of units concerned, set-up times and other uncertainties. Mistakes in the production plan are, however, detected through the simulation and can be corrected.

Each set of product specific rule set is an object of its own, having properties which define the planning strategy of the product. This means that one single rule set (currently containing only twelve rules) can be used for all products, the strategies being completely defined through the values of the properties.

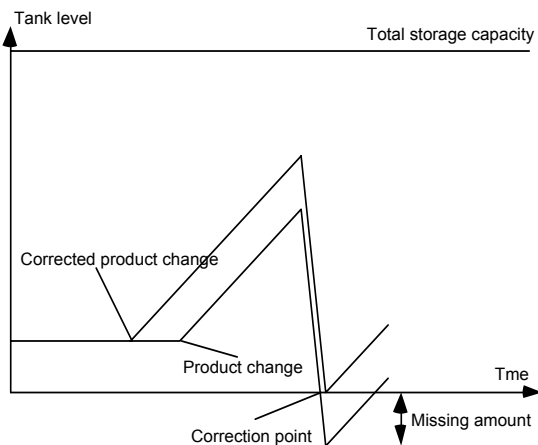


Figure 11. Correcting the production plan.

Quality Tanks:	Min. amount in storage:	<input type="text" value="0"/>	kg	<input type="button" value="OK"/>	
<input checked="" type="checkbox"/> T-002	Min. storage space:	<input type="text" value="000000"/>	kg	<input type="button" value="Cancel"/>	
<input type="checkbox"/> T-003	Time marg. before LST:	<input type="text" value="36"/>	h		
<input type="checkbox"/> T-001	Prod. estimate margin:	<input type="text" value="0"/>	h		
	Backup marg. on empty:	<input type="text" value="000000"/>	kg		
	Time to aim for sec. b.:	<input type="text" value="2160"/>	h		
Heuristics:		Produced:	Not produced:		
Running out:	<input type="text" value="1000"/>	Enough 2:	<input type="text" value="0"/>	Enough 2:	<input type="text" value="1"/>
Going full:	<input type="text" value="0"/>	Not en. 2:	<input type="text" value="950"/>	Lst 2:	<input type="text" value="1000"/>
No batches:	<input type="text" value="1"/>	Enough 1:	<input type="text" value="1"/>	Enough 1:	<input type="text" value="1"/>
Default:	<input type="text" value="1"/>	Not en. 1:	<input type="text" value="950"/>	Lst 1:	<input type="text" value="1000"/>
				No hurry:	<input type="text" value="1"/>

Figure 12. Dialogue for defining the production planning strategy.

The properties of a rule set object are the following:

- Product to which the strategy belongs.
- Tanks to which the product may be produced (unless they already contain another product).
- A set of batches of the product that should be produced.
- The margins one to six.
- Heuristic constants.

Despite its simple structure, the rules express the knowledge of the human production planner very well. Nothing stops the rule base from being bigger or more complicated, but the experiments performed showed that there is no need for this: the twelve rules are already sufficient. The rules are written as a simple "If-Then" structure in Smalltalk, even though a Prolog module is included in Smalltalk/V.

5. RESULTS

The system was tested on several real situations from the past of the phenol plant, and always managed to produce a plan which worked well, and which was quite similar to the one produced by the human planner. This is demonstrated in the example diagrams of figure 13, where the levels of two tanks are shown for a period of two and a half months from the past of the plant. The real used production plan is shown above and the production plan produced by the system is shown below.

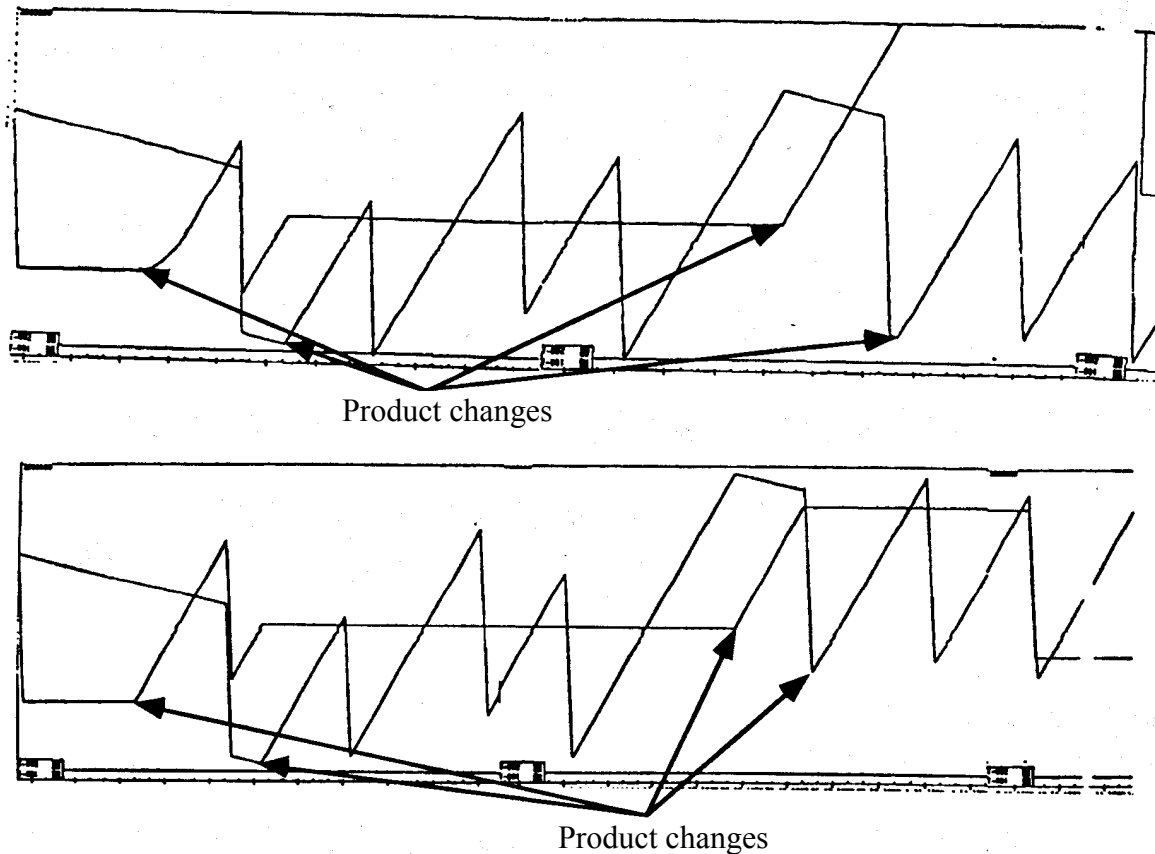


Figure 13. Comparison of production plans produced by the human production planner (above) and the automatic production planning module (below).

The good production plan was found without any corrections with the first try every time once the right margins and heuristic constants had been found. Even with badly chosen margins and constants the correction mechanism always managed to give a working plan in the end.

The methods used should work quite well even in the case of several parallel production units. This aspect is already possible to take into consideration, since the effects of having parallel production units are treated by the simulation. Extending the methods to production other than chemical plants remains to be studied.

6. CONCLUSIONS

The OO approach used has shown several advantages. Managing all the data needed about the plant is simplified significantly by the object approach, which also simplifies the handling of the graphical interface. The programming effort required is also estimated to have been much smaller than the effort required to build an equivalent system with conventional programming techniques.

The graphical interface and object orientation also simplifies the creation and especially the modification of an already existing model to answer to new requirements. The possibility to define hierarchical models adds flexibility to the design since it avoids having to deal with overwhelming quantities of data at the same time. Simulation with the model and information retrieval from it are simplified by the object orientation as well.

The implicit knowledge in an object oriented model also simplifies the use of AI techniques, which is shown by the distributed product specific sets of rules used, which are quite simple but still perform very well.

References

Alasuvanto J., Eloranta E., Fuyuki M., Kida T. Inoue I.: *Object oriented programming in production management - two pilot systems*, Int. J. Prod. Res. 1988, vol.26, no. 5, p. 765-776.

Falster P.: *Planning and controlling production systems combining simulation and expert systems*, Computers in Industry, nr.8 1987, p. 161-172.

Främling, Kary: *Using object oriented models for the scheduling of production in petrochemical industries*, MSc. thesis : Technical University of Helsinki, 1990. 86 p.

Hammarström Leif: *Dynamic Simulation of Chemical Engineering Processes by using Object Oriented Programming*, Neste Oy, Process Automation, Kullo, Finland 1990. 31 p.

Henning Gabriela P., Leone Horacio P., Stephanopoulos George: *Hierarchical Modelling Environment for Process Engineering*, Laboratory for Intelligent Systems in Process Engineering Technical Report, Massachusetts Institute of Technology, Cambridge, MA 02139, 1988. 14 p.

Jensen Paul A.: *Operations Research*, Holden-Day, Inc., Oakland CA 1986. 502 p.

Kramer Mark A.: *Expert Systems for Process Fault Diagnosis: A General Framework*, Department of Chemical Engineering Technical Report, Massachusetts Institute of Technology, Cambridge, MA 02139, 1987. 31 p.

Kramer Mark A.: *Automated diagnosis of malfunctions based on object-oriented programming*, Lipse Technical Report, Massachusetts Institute of Technology, Cambridge, MA 02139, April 1988. 28 p.

Kramer Mark A., Finch Eric F.: *Fault diagnosis of chemical processes*, Department of Chemical Engineering Technical Report, Massachusetts Institute of Technology, Cambridge, MA 02139, September 1987. 31 p.

Lakshmanan Ramachandran, Stephanopoulos George: *Synthesis of Operating Procedures for Complete Chemical Plants, Part I: Hierarchical, Structured Modelling for Nonlinear Planning*, Lipse Technical Report, Massachusetts Institute of Technology, Cambridge, MA 02139, February 1988. 51 p.

Realf Matthew: *Learning localized heuristics in batch scheduling*, Lipse Technical Report, Department of Chemical Engineering, Massachusetts Institute of Technology. May 1989. 37 p.

Realf Matthew: *An analysis of the chemical batch production problem and a detailed methodology for scheduling*, Lipse Technical Report, Department of Chemical Engineering, Massachusetts Institute of Technology. May 1989. 46 p.

Realf M. J., Stephanopoulos G.: *Machine Learning for the Improvement of Scheduling Algorithms*, Lipse Technical Report, Massachusetts Institute of Technology 1990. 26 p.

Stephanopoulos George, Johnston James, Lakshmanan Rama: *DESIGN-KIT: An Intelligent System for Planning Plant-wide Process Control Strategies*, Lipse Technical Report, Massachusetts Institute of Technology, Cambridge, MA 02139, March 1987. 10 p.

Stephanopoulos George: *The Scope of Artificial Intelligence in Plant-wide Operations*, Lisper Technical Report, Massachusetts Institute of Technology, Cambridge, MA 02139, September 1987. 46 p.

Stephanopoulos G., Johnston J., Kriticos T., Lakshmanan R., Mavrovouniotis M., Siletti C.: *DESIGN-KIT: An object-oriented environment for process engineering*, Comput. chem. Engng. Vol. 11, No 6 1987, pp. 655-674.

Stephanopoulos G.: *Artificial Intelligence... 'What will its contributions be to process control?'*, Lisper Technical Report, Massachusetts Institute of Technology, Cambridge, MA 02139, December 1988. 69 p.