INTEGRATION OF *MATHEMATICA* INTO AN INFORMATION SYSTEM SUPPORTING CHEMICAL PROCESS DEVELOPMENT

Juha R. Anttila & Antti P. Pasanen*
Laboratory of Chemical Process Engineering, Department of Process Engineering, University of Oulu, Finland
**Technical Research Centre of Finland, Chemical Technology*

ABSTRACT

Mathematica has been used as an environment for object modelling of a new chemical pulping process. The usability of *Mathematica* has been enhanced by integrating it into an Information System (IS) supporting chemical process development. The implementation of IS is based on a new phenomenon driven process design methodology. The integration into the IS results in an easy way to change and give process structures for *Mathematica* and makes *Mathematica* an attractive alternative compared to commercial simulators.

1. INTRODUCTION

Process development is composed of prescriptive modelling of the process to be developed and the associated management activity. In case of chemical processes modelling can be viewed as 1) making mathematical models, which refer to algebraic and differential equations describing phenomenological laws and the laws of conservation and 2) describing process structure, which is a result of process synthesis activity. In order to enhance process development activity a systematic procedure to guide both the process modelling and the management activity is needed. For this purpose a new Phenomenon Driven Process Design (PDPD) methodology has been developed [1].

In our previous work [2] we reported the use of *Mathematica* as an environment for object-oriented modelling of chemical processes. The input of process structure was found to be cumbersome because process structure is given to *Mathematica* by matrices and lists that had to be manipulated manually. In this work we integrate *Mathematica* into an information system that supports chemical process modelling. By means of this integration the input of process structure can be done by using a graphical interface, which results in more efficient use of *Mathematica.*

2. PHENONOMENON DRIVEN PROCESS DESIGN METHODOLOGY (PDPD)

PDPD methodology provides procedural guidance for how to start and proceed with design projects, and how to develop and model chemical processes. The use of this methodology for supporting process development is expected to improve the quality of design projects by offering a framework for enhanced communication, for creative solutions and for documentation of all the relevant knowledge. Chemical process development with PDPD is based on two methodological definitions: (1) "Design project is control of resources for a purpose" and (2) "Chemical process is

control of physico-chemical phenomena for a purpose". These definitions decompose into further guidelines how to set goals, how to allocate resources and how to use PDPD's modelling elements in conceptual design. According to PDPD methodology a chemical process development project is viewed as a design cycle which is shown in Figure 1.

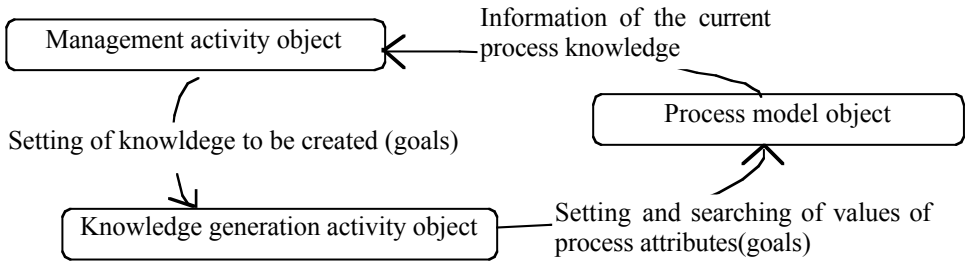

Figure 1. Design cycle.

2.1. PSSP formalism

The concept system behind the methodology - called PSSP language - represent any real thing as an object with four attributes, Purpose, Structure, State, and Performance (PSSP), resulting in a transparent object hierarchy with a highly unified format. The structure of any object (see Generic Real Thing in Figure 2) has two dimensions, one for disaggregating an object topologically relative to similar objects, such as disaggregating a project to be composed of sub-projects or a chemical process to be composed of sub-processes. The other dimension is unit-structure for linking an object to be composed of objects from other classes; see indentation in Figure 2.

The models describing the behaviour of a process are located under the State attribute of 'Chemical process'. The state models relate the values of State variables to time and to other variables; StateVar = f(x,t). These mathematical expressions are meant to capture the Purpose, Structure, State and Performance from mathematical models, such as mass and energy balances. PDPD methodology is not supposed to do the mathematical solving by it self, but to enable to link PDPD-formatted knowledge to 'External' applications, where the actual numerical manipulations are performed.

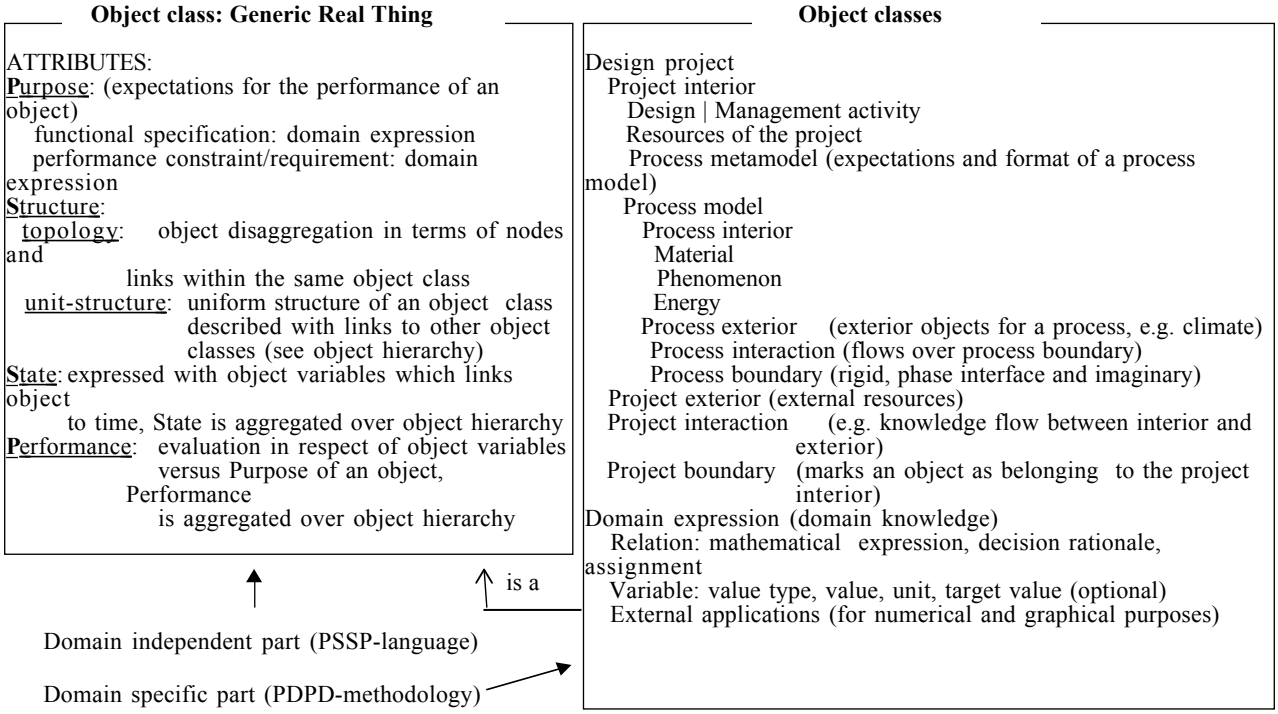

Figure 2. Concepts of PDPD methodology

The procedural guidance how to use the modelling elements consist of two parts; a) Performance driven strategy and b) Phenomenon driven ordering.

2.2. Performance driven strategy

Performance driven strategy is depicted in Figure 3 [3]. The main idea is to express the Purpose for every object and, after that, to specify the Structure and the State of the object. It is then possible to evaluate the Performance (or goodness) of the object and to make design decisions on the object's Structure, State or even Purpose. The Purpose is given with a set of Purpose expressions, according to which the Performance is evaluated continuously throughout the lifetime of the object.

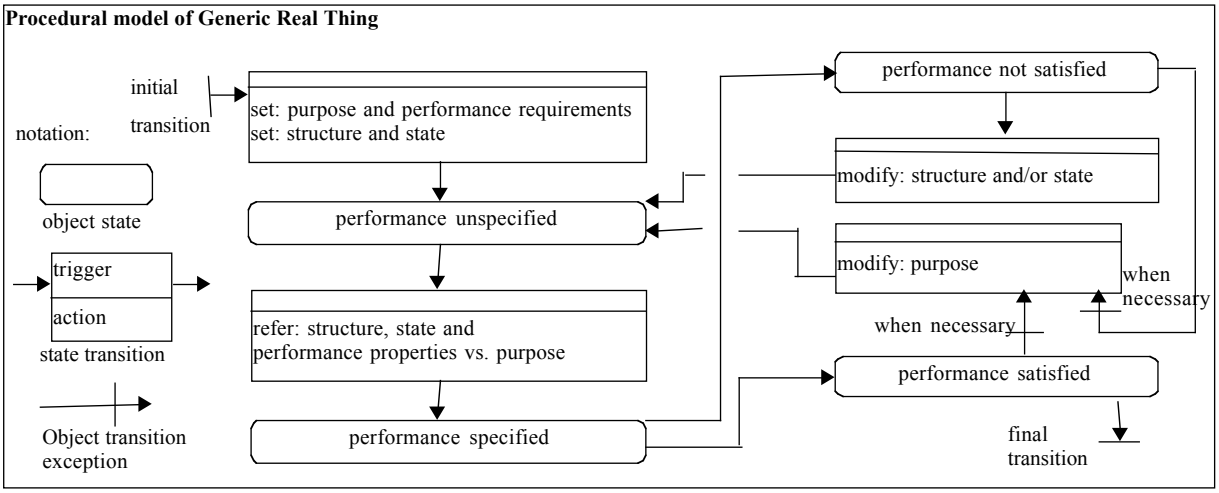

Figure 3. Performance driven strategy to edit objects.

2.3. Phenomenon driven ordering

Phenomenon driven ordering originates from definitions 1 and 2. It means that a development project should aim to reveal how to control the phenomena and material involved in the process. This necessitates to find out: (1) what is the feasible state-distribution of material and phenomena that possibly results the desired process performance, and (2) how to create and maintain this state-distribution. The means to create and maintain the desired state-distribution(s) are: selection of necessary and auxiliary phenomena and material, selection of process boundary, and setting of process interactions (flows) and, if necessary, disaggregation of phenomena into separate sub-processes. According to PDPD, disaggregation should be considered only if there are no feasible unit-structural modifications producing the desired state-distribution and process performance.

3. IMPLEMENTATION OF PDPD AS AN INFORMATION SYSTEM

The use of PDPD implies computer support and this has been realized as an Information System (IS). The developed tool provides a generic representation of both the chemical processes to be designed and the activities producing the process model. PDPD-IS is implemented in an object-oriented environment, MetaEdit+®. The use of PDPD-IS results in integration of knowledge via a unified concept system and in thorough documentation of the design project. This tool also makes it possible to link PDPD formatted knowledge representation to external software environments. In chapters 3.1 and 3.2 a brief introduction to PDPD-IS is given.

3.1. Interface for using the modelling elements

The implementation procedure of PDPD with MetaEdit+ is fairly straightforward and provides fast prototyping. With MetaEdit's metamodelling language the classes, class properties and property inheritance (see Figure 2) are set up. This stage also includes symbolisation of the modelling elements and setting up of dialogues for a PDPD user. The object-hierarchy guides PDPD users to instantiate and to view objects in a particular order with a graphical editor. These so called metalevel definitions ensure that engineering data and knowledge are represented in PDPD-format. The outlook of the system is shown in Figure 4. In order to check or process the content of the object-repository MetaEdit+ provides a query language. For a PDPD user we have programmed automated data processes:

- Report generation of the object repository
- Generation of suggestions according to Phenomenon Driven ordering and Performance strategy.
- Transcription of PDPD-formatted process models to external applications (input-file generation); for example *Mathematica* or ASPEN+

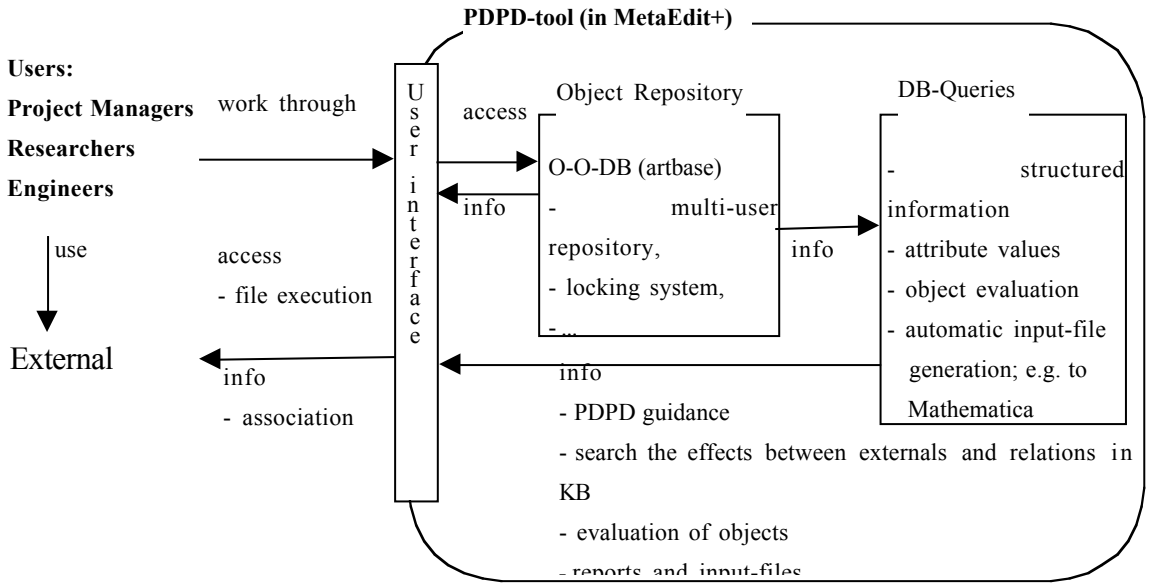

Figure 4. The implemented system description from user's perspective.

3.2. Linking ability

The last point in chapter 3.1; " Transcription of PDPD-formatted process models to external applications", provides a way to link PDPD-formatted data to external software environments. MetaEdit+ also enables to associate (execute) external software environments. This feature is also

applied in PDPD, as can be seen in Figure 4. Objects and their properties can be associated with external software environments. These external environments might be databases, numerical solvers, graphical representations, and so on. Thus PDPD-IS serves an information integrator connected to the object-repository. This object repository in turn is constructed to follow PDPD methodology.

In regard to PDPD-IS *Mathematica* is an external application. Before use of *Mathematica* the structural description of the process in MetaEdit+ must translated into an expression that can be utilized in *Mathematica.*

4. EXPRESSION OF PROCESS STRUCTURE

In MetaEdit+ a graphical diagram editor is used for instantiation of process units and process flows. Specification and viewing the properties of process units and flows are carried out through dialogue boxes. Process interior (material, phenomenon and energy) can be associated with variables describing the state of processes and flows. A variable can be assigned (fixed) to a certain value (constraint) or it might be a variable to be solved. The value of a variable might also be associated to a mathematical expression solving the variable (or set of variables). These values of variables or mathematical expressions, say mass balance equations, can be linked to external software environments. An example of process structure and associated state models is shown in Figure 5.

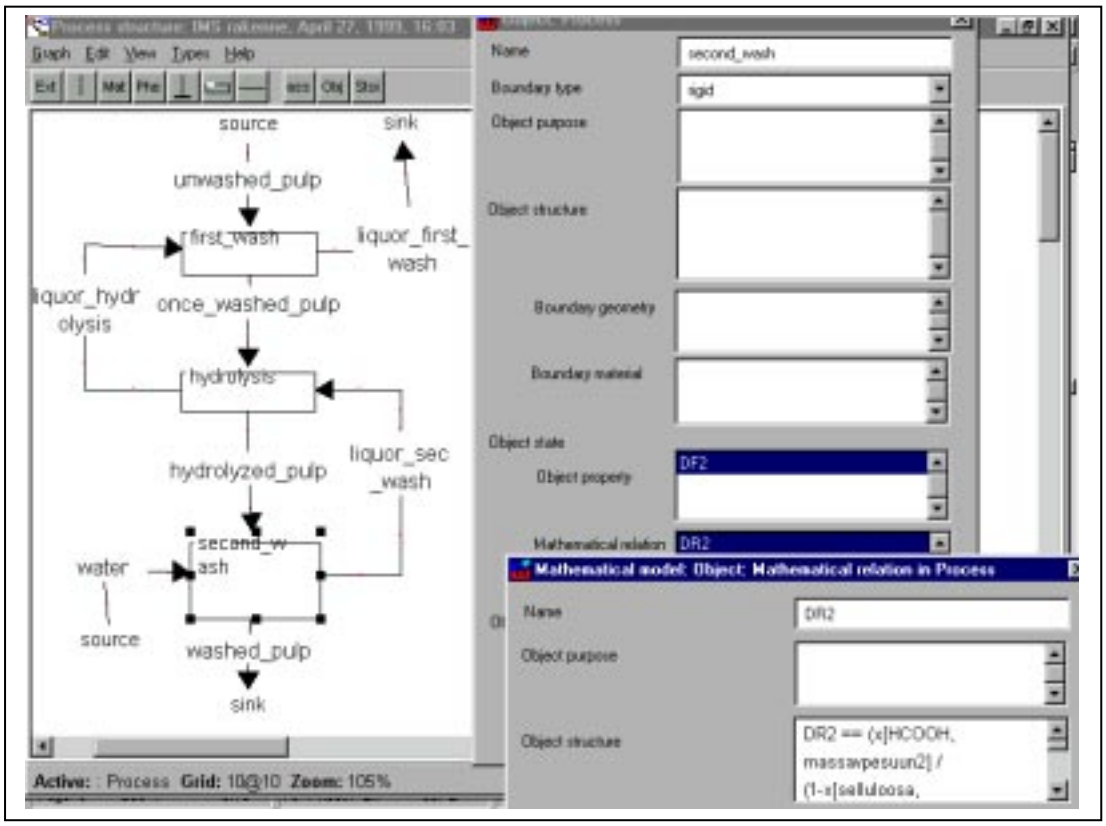

Figure 5. A process structure (detailed topology) and dialogue boxes for one process unit.

MetaEdit+ supplies a query language that can be applied for generating input-files from the object repository (see Figure 4) to a external software. We have programmed a query generating an expression that can be used as an input-file in *Mathematica*. The query goes through all the process units, their connections (flows) and the mathematical expressions involved (assignments and relations). The user runs the query in MetaEdit+ and then applies it *in Mathematica*. Figure 6 shows a part of the programmed query language and an example of the resulted *Mathematica* input-file. This part of the query goes through the origin of every flow and transcribes it into a *Mathematica* expression.

```
'where_fromF={';
foreach .Process
   {'{';
   dowhile ~From>Object interaction;
      {
      foreach >Object interaction
         {
         if id;=id;1; then '1'; ','; else '0'; ',';endif;
         }; '}'; '+'; '{';
      }; '},'; newline;
   }; '};'; newline;
'where_fromF = pos[Transpose[where_fromF]];';
newline; newline;
```

```
where_fromF
={{0,0,1,0,0,0,0,0,}+{1,0,0,0,0,0,0,0,},
{0,0,0,1,0,0,0,0,}+{0,0,0,0,0,0,1,0,},
{0,0,0,0,1,0,0,0,}+{0,1,0,0,0,0,0,0,},
};
where_fromF = pos[Transpose[where_fromF]];
```

Figure 6. A part of the programmed query language (left) and its result (right).

5. PROCESS ANALYSIS AND EVALUATION

In our previous work [2] we reported the use of *Mathematica*: (1) as an environment for object-oriented modelling of structural and behavioral features of a chemical process and (2) a symbolic and numerical equation solver. The program that we have written for *Mathematica* takes the structural description of the process by matrices and lists as an input and gives the process state model, based on mass balances, as an output. These equations are then solved by using *NSolve* command. The use of *Mathematica* is thus a part of process design cycle (Figure 1) and can be illustrated as in Figure 7A. The process design cycle can also be illustrated as an object flow diagram as in Figure 7B, which is an expression of the performance driven strategy (Figure 3) in case of chemical processes.

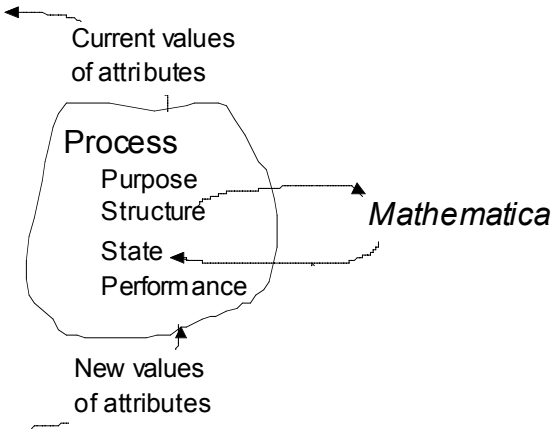

Figure 7A. Use of *Mathematica* as a part of process design cycle.

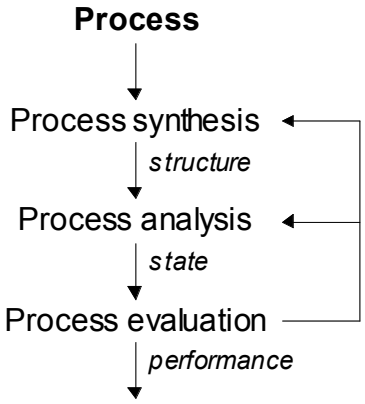

Figure 7B. Process design cycle as an object flow diagram.

In Figure 7B process synthesis refers to modelling of process structure and process analysis refers to applying algebraic and differential equations describing phenomenological laws and the laws of conservation for specifying the process state. Thus, with respect to process design cycle *Mathematica* is used as a tool to automatically perform process analysis. Process performance is the property on the basis of which the assessment of real process feasibility is made. Its value follows from the values of structure and state attributes and thus process performance can be influenced by giving those attributes different values. If process performance does not fulfill the purpose, process structure and/or state will be altered.

6. DEVELOPMENT OF A NEW CHEMICAL PULPING PROCESS

In this work PDPD-IS has been used as a tool to support the development of a new commercial chemical pulping process. The process uses formic acid as pulping chemical and pulp is bleached by using oxygen chemicals [4]. Process is environmentally more friendly than conventional pulping processes because the process does not utilize chlorine nor sulphur chemicals and because the process causes very little emissions to air or to water. The process can be used for producing pulp from non-wood raw materials and the produced pulp can be used for example for producing fine papers.

The performed computations in this work are similar to those done in our previous work. *Mathematica* is used as an computational tool to search the optimal structure for the process. This search has been done by experimenting topological alternatives by material allocation and by studying the effect of variation of different state variables. In terms of PDPD, process structure and state have been altered and the effect on process performance has been monitored. As an example Figure 5 shows a process structure for which the mass balances have been evaluated. This process is a part of the actual formic acid pulping process in which the formic acid is removed from pulp by washing with water. For example, the effect of dilution factor on the amount formic acid remaining in the pulp after the second wash has been evaluated.

7. DISCUSSION

7.1. Advantages of the approach compared to commercial simulators

There are numerous sophisticated commercial simulation programs available and PDPD-tool could be integrated into these programs as well as into *Mathematica*. However, these programs do not allow open problem formulation, which means that without certain input, say temperature of a flow, these programs can not evaluate mass balances. Furthermore, in these programs the process structure has to be given by combining pre-programmed unit processes. These limitations decrease the degrees of freedom in modelling work and may prevent finding new creative solutions.

The approach implemented in *Mathematica* does not limit the freedom of modelling work and thus it can be viewed to support creative process design better than commercial simulators. It should be noted that this advantage of *Mathematica* does not require integration of *Mathematica* to other programs. Software integration renders modelling work more efficient but it does not immediately enhance the creativity of design work.

7.2. Advantages of the software integration

The structural description of the process is given to *Mathematica* by matrices and lists which are quite clumsy to use if the process structure should be changed. In this work these matrices and lists are generated automatically on the basis of PDPD formatted representation, which has been generated by using a graphical interface. As a result of this, changing or giving an entirely new process structure is much easier compared to an alternative, where matrices and lists should be generated and manipulated manually. Thus, regarding the use of *Mathematica* the greatest advantage of software integration is the ease and the speed of process analysis.

One advantage of commercial simulators compared to *Mathematica* is that they have a graphical interface for the input of process structure. The software integration presented in this paper cuts down this advantage and makes the use of *Mathematica* an attractive alternative compared to commercial simulators.

7.3. Deficiencies of the approach

Mass balances are a set of nonlinear algebraic equations. Generation of these mass balance equations was always successful and took a relatively short time, but in some cases the solution of the model caused problems. Obviously it is the algorithm behind the *NSolve* command which is ineffective. We noticed this problem already in our previous work. However, *NSolve* seems to work more effectively in *Mathematica's* version 3.0 than in older version 2.2, but it would be desirable to develop the command further.

8. REFERENCES

1. "Pohjola, V.J. & Tanskanen, J., Phenomenon Driven Process Design Methodology: Formal Representation" given at the Session "Design, Synthesis, Management" of CHISA'98 Conference, Aug. 23-28, 1998, Prague, Czech Republic.

2. Pohjola, V.J. & Anttila, J.R. Chemical process synthesis: Object modelling and automatic model generation. International *Mathematica* Symposium, IMS'97. Rovaniemi, 29.6.-4.7.1997.

3. Embley, D.W., Kurtz, B..D. & Woodfield, S.N. Object-Oriented Systems Analysis: A model-Driven Approach. 1992 New Jersey. Yourdon Press Computing series, Prentice Hall.

4. Rousu, P.P. & Rousu, P. High Grade Nonwood Based Paper Economically and Environmentally Friendly. 1997 Pulping Conference, San Francisco, October 19-23 1997. Atlanta 1997. TAPPI. Book 2, pp. 487-492.