Modelsaz: An Object-Oriented Computer-Aided Modeling Environment

Farzi , Ali and Mehrabani, Arjomand *[*+]

Department of Chemical Engineering, Isfahan University of Technology, Zip code 84156, Isfahan, I.R. IRAN

ABSTRACT: *Modeling and simulation of processing plants are widely used in industry. Construction of a mathematical model for a plant is a time-consuming and error-prone task. In light of extensive advancements in computer science (both hardware and software), computers are becoming a necessary instrument in industrial activities. Many software tools for modeling, simulation and optimization of processing plants have been developed. In this paper, a new tool, called "Modelsaz", for the modeling of physical-chemical-biological processing systems is introduced. This software can automatically generate mathematical models of various processing plants in Mathematica™ compatible format, based on conservation principle. Lumped systems are the basic elements of processing plants. The dynamic lumped models of the plants are based on transient mass and energy balances of these basic elements. "Modelsaz" has been developed under Microsoft-Windows 2000 operating system using Microsoft Visual C++6 programming environment. The software uses object-oriented features and has a user-friendly graphical interface in order to facilitate the construction, modification and reusability of a processing system models.*

KEYWORDS: *Object-oriented modeling, Simulation, Physical topology, Species topology, Modelsaz*

INTRODUCTION

Environmental and safety regulations, increasing demands for product quality and growing competitions in products marketing, necessitate continuous improvement of chemical plants within minimum time and costs. Availability of models of processing systems with varying degrees of details can decrease the number of required experiments. While, crude models are sufficient for the evaluation of process alternatives in early stages of design, detailed models are required for further works on selected flowsheets. Therefore, a set of models with varying degrees of details is required for each processing unit. Generation of these model families and saving them for the next applications is a great task and must be supported by the information technology.

This paper introduces *Modelsaz* as a new software tool for the modeling of processing plants. By using this software, it is possible to quickly create a new processing plant and its mathematical model. This model consists of

** To whom correspondence should be addressed.*
+ E-mail: Arjomand@ cc.iut.ac.ir
1021-9986/04/1/35 16/$/3.6

a set of dynamic mass and energy balance equations, which are first-order ordinary differential equations with respect to time. These equations are generated in *Mathematica*™ compatible format and can be solved by it [1]. Results can be plotted for better assessment of modeled plant.

Methods used by the current simulation tools can be divided into two major groups [2],[3]:

- Block-oriented (modular) methods
- Equation-oriented method

In both groups, either sequential or simultaneous solution algorithms may be used [2]. In block-oriented approach, each unit is shown by a block diagram, which models the unit behavior. All these blocks are linked to each other by connections in order to represent the flow of mass, energy and information. The models of these units have been generated before, by modeling experts and stored in a model library for future uses.

In this method, the implemented chemical engineering knowledge and the model structure are fixed and cannot be accessed. Only models of physical properties can be selected independent of unit models.

Equation-oriented simulation tools support both use of predefined models and definition of new models based on declarative modeling tools. No other tool exists for the assistance of experts or end users in generation of new models. Therefore, modeling at level of units, requires broad range of knowledge in several contexts such as chemical engineering, modeling and simulation, numerical mathematics, and computer science. So, generation of new plant models by this type of tools is often limited to small groups of experts.

Block-oriented approach, simply and effectively helps engineers in solving standard flowsheets, but does not sufficiently support the solution of more complex problems. This issue is largely due to requirement to models of various processing units, such as multi-phase reactors, membrane processing units, polymeric reactors and solid handling units with varying degree of details. Thus generation and development of models is often expensive and time-consuming. Equation-oriented languages support the application of the models, but nor provide assisting tools for helping users in generation of models from engineering concepts, neither support storing and reusing of them. Thus, unessential and extra modeling effort for similar units is unavoidable, which consequently will cause loss of time and money.

Modeling can be done by the implementation of modeling languages, [4 to 15], or graphical environments [16]. The aim of all the modeling tools is to assist and simplify generation, development, saving and maintenance of the models in a consistent form. Some of the modeling and simulation tools have been integrated in order to support modeling, simulation and analysis tasks for design, control and operation of processing plants. For example, OMOLA modeling language and OMSIM simulator have been integrated as a control system design environment [18]. Some new modeling tools can be shown as Table 1, [17].

MODELING METHODOLOGY

Construction and development of any computer application must support the following concepts in order to have a consistent modeling methodology:

1) Decomposition of models and definition of elementary modeling objects as building blocks of a consistent model,
2) Defining general modeling procedures, which must primarily support both modeling aspects of reuse and modification of an existing model to meet a new requirement.

In this work, the natural structure of processing plants is used for the definition of basic modeling objects.

STRUCTURAL DESCRIPTION

The structure of a processing plant can be divided into three major parts:

Table 1: Some new modeling tools [17]

Modeling	Integrated with simulator	Reference
ASCEND	*	[4]
DYMOLA		[5]
DYLAN	*	[6]
GPROMS	*	[7]
HPT		[8]
MODASS		[9]
MODEL.LA		[10]
MODELLER		[11]
MODEX		[12]
OMOLA	*	[13]
PROFIT		[14]
VEDA	*	[15]

Physical Structure

Physical structure of a plant consists of various parts. A plant unit can be decomposed to its constituent parts until non-decomposable parts are achieved. These non-decomposable parts are really or virtually considered to be homogeneous and called *simple systems*. For example, a distillation column can be decomposed to its constitutive parts such as trays, reboiler and condenser. Each of these parts can be decomposed further; e.g., liquid and gas phases can be considered to be sub-systems of the trays, which cannot be decomposed, further. Fig. 1 demonstrates such a decomposition. Decomposition of a plant results in a hierarchy of systems that can be shown in the form of a tree structure [11], [19]. The resulting tree structure is strictly hierarchical, i.e., any node in the tree structure has only one parent. This issue is a natural result of application of conservation law as the main basis for the mathematical representation of processing plants. As an example, consider a tree structure as represented in Fig. 2.

Mass balance equation for node p is as follows:

$$M_P = M_{P1} + M_{P2} \tag{1}$$

Similarly for nodes p_1 and p_2 we have:

$$M_{P1} = M_{P1,1} + M_{P1,2} + M_{P1,3} \tag{2}$$

$$M_{P2} = M_{P2,1} + M_{P1,3} \tag{3}$$

Replacing last equations in equation (1) results:

$$\begin{aligned} M_P &= (M_{P1,1} + M_{P1,2} + M_{P1,3}) + (M_{P2,1} + M_{P1,3}) \\ &= M_{P1,1} + M_{P1,2} + 2M_{P1,3} + M_{P2,1}) \end{aligned} \tag{4}$$

Thus node $p_{1,3}$ is accounted two times, which violates conservation law. So the tree structure of systems must be strictly hierarchical.

The plant itself is the root node of the tree structure and simple systems are its leaves. Each node in a tree structure and its child nodes are represented mathematically as follows [19], [20]:

$$\begin{aligned} P &= [s]^u \\ B_P &= \{m \mid u_m \equiv s_p\} \end{aligned} \tag{5}$$

Where, p is the node, s is its identifier and u is its parent identifier. B_p is the set of all nodes, whose parent

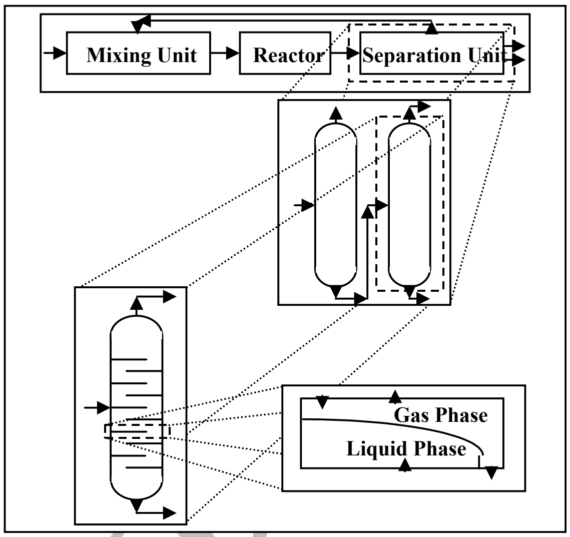


Fig. 1: Demonstration of a processing plant and an alternative decomposition

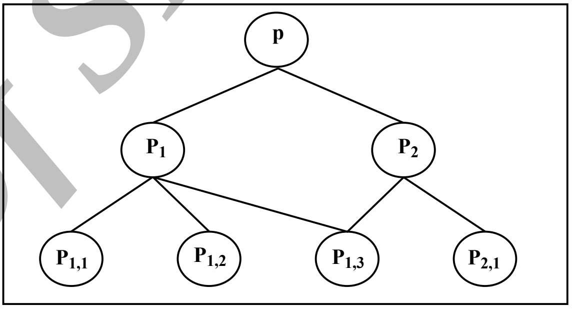


Fig. 2: A tree structure that is not strictly hierarchical

identifier is equal to the identifier of node p. The set of all nodes in a given level of a tree structure can be calculated.

$$\begin{aligned} p_{root} &= [1]^0 \\ P_1 &= \{p_{root}\} = \{[1]^0\} \\ P_d &= \bigcup B_p \Big| \forall_p \in P_{d-1} \qquad d = 2,3,\ldots,\max(d) \end{aligned} \tag{6}$$

As shown, parent identifier of root node, p_{root}, is set to zero. P_d is the set of all nodes at level d of the tree structure. Finally the tree structure can be shown in the following form:

$$\tau = \bigcup_{d=1}^{\max(d)} P_d \tag{7}$$

In the above equation, τ is the set of all nodes of the tree structure and max(d) is the maximum possible level of it. Fig. 3. shows a simple tree structure.

Some operations can be done on a tree structure such as adding, removing and moving a tree structure and also adding an intermediate node to or deleting an intermediate node from each position of the tree structure. All these operations are described in detail and represented mathematically [19], [20]. Only adding a new tree structure to an existing one is represented here as a sample.

Any number of tree structures can be added to a tree structure. Suppose the addition of tree, τ^2, which β is its root node, to tree, τ^1, under node α. By default, any new node will be added to the right of node α. Parent identifier of node β is changed to the identifier of node α, so this node becomes a child node of α.

$$u_{\beta}^{2} = s_{\alpha}^{1}$$
$$B_{\alpha} = B_{\alpha}^{1} \bigcup \{\beta\}$$
$$P_d = P_d^1 \bigcup P_{d-\ell}^2 \qquad d = \ell+1, \ell+2, ..., \max(d) \tag{8}$$

Finally, the node and parent identifiers of tree, τ^2, are modified based on the following algorithm:

$$i = s_{\gamma}$$
$$\forall p \in P_d; (\forall m \in B_p; u_m = i+1); \; s_p = i+1; i = i+1 \tag{9}$$

Where, s_γ is the last node to the right of node αat level level ℓ, which is currently is node α itself. Fig. 4 shows this operation.

For handling the physical structure of a process, a C++ class, named *System*, was defined. This class has attributes such as system name, system identifier, number of sub-systems and pointer to the super-system. As discussed above, every system of a processing plant is located in a hierarchical structure, which is indeed a tree structure of the systems. A class called *DoubleList*, dynamically handles the tree structure and the class *System* has inherited its attributes and operations.

Network of systems and connections

Connections show the communication paths between systems of a processing plant. Systems can communicate mass, energy, information and other types of quantities with each other. In this work, only types of connections, e.g. mass, heat and work connections have been used. Both ends of connections must be connected to simple

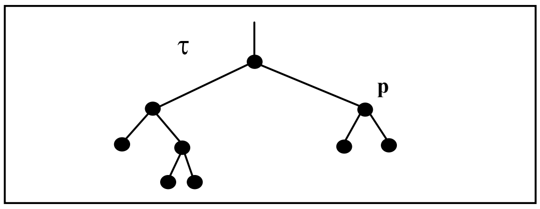


Fig. 3: A simple tree structure

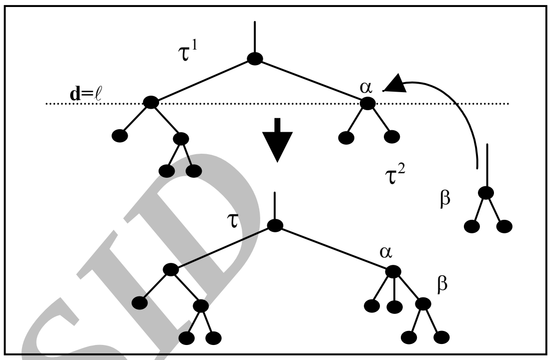


Fig. 4: Adding a tree structure

systems in order to maintain the consistency of the plant's structure. Therefore connections cannot be defined in such a way that one or both ends left free. The set of connections between simple systems can be represented formally as [11]:

$$C = \left\{ c_{a|b} \middle| \forall a, b \in S^s \right\} \tag{10}$$

Where, S^s is the set of all simple systems in a plant and $c_{a|b}$ is a connection between systems a and b.

Connecting simple systems to each other, results in a *network* of all simple systems and connections[11], [19], [21]. It is not necessary all simple systems in a network to be connected. By considering direction for connections, the network is called *directed network*. Sub-nets are subsets of base network that consist of some simple systems and some connections, but the boundary of a sub-net cannot cut any connection path. Typed networks can also be identified by including all simple systems and connections with specified type. Mass and heat transfer networks, are examples of typed networks. If all systems in a network are connected to each other, the network is called a *domain*. Similarly typed domains can be specified. Collection of physical structure and connections is called *physical topology* [11], [19].

The class *Connection* handles all of the connections existing in a plant. This class has necessary attributes and operations for creating connections such as pointers to source and target systems and the type and direction of a connection. This class has a restriction that source and target systems of any connection must be simple systems. Two pointers have implemented in the definition of *Connection* class for accessing to source and target systems of a connection.

Species Distribution or Species Topology

Obviously all systems may not have the same species. For example, in a shell and tube heat exchanger, species in shell side and tube side usually are not the same. Also in the presence of a chemical reaction in a system, produced species may or may not exist in other systems. Furthermore, in membrane processes a mass connection can act selectively on the transfer of species and not allow some specific species pass through. Therefore, some basics and algorithms for the computation of species distribution between systems are essential. Initially two main operators are defined [11]. *Permeability operator* determines which species cannot permit from a mass connection. The actual set of species that can pass through the connection depends on the species sets of the connected systems.

$$\mathcal{A}_{a|b} = \mathrm{P}(\mathcal{A}_a \cup \mathcal{A}_b) = \mathcal{A}_{pc} \cap (\mathcal{A}_a \cup \mathcal{A}_b) \qquad (11)$$

Where,

P :: permeability operator,

$\mathcal{A}_{a|b}$ :: set of all transferable species between systems a and b,

$\mathcal{A}_a$, $\mathcal{A}_b$ :: set of species of systems a and b,

A_{Pc} :: set of species that can pass through connection c.

Also *reactivity operator* is defined to show potential species that can be produced in presence of reactants:

$$\mathrm{R} = \left[\left\{ \mathcal{A}_r^R \right\} \longrightarrow \left\{ \mathcal{A}_r^P \right\} \right] \qquad (12)$$

The operation of this operator can be described mathematically as:

$$\mathrm{R}(\mathcal{A}) = \begin{cases} \mathcal{A}_r^P & \textit{if} \quad \mathcal{A}_r^R \subseteq \mathcal{A} \\ \varnothing & \textit{else} \end{cases} \qquad (13)$$

Where,

$\mathcal{A}$ :: set of all species in a plant,

$\mathcal{A}_r^R$ :: set of reactants of reaction r,

$\mathcal{A}_r^P$ :: set of products of reaction r.

There are three different sources of species in a system:

1) Injection of species by users
2) Communication through connections
3) Production by reactions

Two algorithms, namely *Coloring* and *Add Species*, which use *permeability* and *reactivity* operators, are developed for the construction of species topology. Initially *Coloring* algorithm determines the domain of each species, which can propagate in the plant, based on above operators. Then, *Add Species* algorithm injects species into the systems by implementing the domains specified by *Coloring* algorithm [19].

Several classes have been defined for management, construction and alteration of species topology. For example, the class *SpeciesDataBase* handles the database for the components and their physical and chemical properties, the class *Reaction* is used for managing chemical reactions database, and the class *Species* has been created for the construction of species topology by the aid of above two classes. Furthermore, for mass connections, an additional attribute, namely *permeability*, has been defined. This attribute is used for simulating semi-permeable walls such as membranes.

Behavioral Description

Each element in the physical and species topology is specified by a number of properties. These properties are used for generating balance equations of the systems. The behavior of a system is defined by its governing equations. For a system, conservation law is the fundamental basis for the definition of its behavioral description. Some of simulators has been developed based on steady-state balance equations. These simulators cannot cope with operations such as startup and shutdown of a process, or safety and hazard analysis. Also process control lies on dynamic balance equations. So dynamic models are the most general form that can be used for a wide range of activities. In fact, static model is a specific form of dynamic one that its state variables do not change with time.

As simple systems are building blocks of processing plants, the model of a plant can be created by the generation of balance equations for all simple systems. These balance equations are ordinary differential equations with respect to time and consist of total mass balance, component mass balance and energy balance equations. Similar to the decomposition of the physical structure of a plant, implemented quantities in the equations can be defined hierarchically by other relations. This increases the degree of details sequentially. Balance equations of extensive quantities are the topmost level of this hierarchy. Different laws specify the amount or rate of each quantity. These laws are semi-empirical physical-chemical relations or experimental correlations and are located after balance equations in the hierarchy. Despite of the physical structure, this structure is not necessarily strictly hierarchical. According to Fig. 5, variables of the balance equations are decomposed to a set of relations. Also quantities in equations of state and phenomenological coefficients can be refined to their elementary ones.

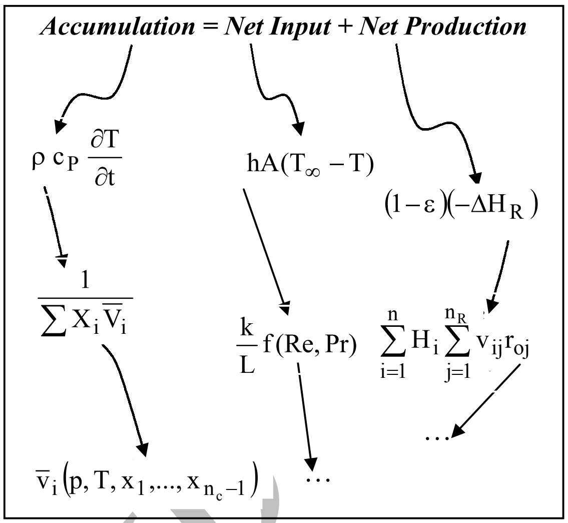
Finally primary state variables such as temperature, pressure and mass fractions of each component in every simple system will be accessed at the lowest level of the hierarchy. Any of the classes introduced above, have some attributes and operations for managing the creation of model equations. For example, codes of equations for transfer laws can be stored in *Connection* objects, while codes for equations of variable transformation are accessible from the *System* objects. Also kinetic laws are introduced by a class, named *Kinetics*. This class has close interactions with the class *Reaction*.

Implementation of Concepts

All of the mentioned concepts are considered as a foundation for the construction of a new software tool for modeling and simulation of physical-chemical-biological systems [19]. This software, namely *Modelsaz*, is developed using Microsoft Visual C++ [22] under Microsoft Windows 2000 operating system.

Software Description

Modelsaz interface is a user-friendly graphical environment. Construction and development of process models can be done easily and quickly. The interface utilizes a menu bar to access most of the operations that can be done on a plant; a standard toolbar for direct



Accumulation = Net Input + Net Production
$\rho c_P \frac{\partial T}{\partial t}$
$hA(T_\infty - T)$
$(1-\varepsilon)(-\Delta H_R)$
$\frac{1}{\sum X_i \bar{V}_i}$
$\frac{k}{L} f(Re, Pr)$
$\sum_{i=1}^{n} H_i \sum_{j=1}^{n_R} v_{ij} r_{oj}$
$\bar{v}_i(p, T, x_1, ..., x_{n_c-1})$
…
…

Fig. 5: Energy balance equation for a reactor and a possible decomposition of its quantities

access to some of the most frequently used commands of the menu bar; a unit-operation toolbar for quick creation of some most commonly used unit operations; and a status bar at the bottom of the application window for showing some hints about a selected command. By clicking the right mouse button in any part of the window a pop-up menu will appear, which shows some of the available operations. The commands of this pop-up menu depend on the position of the mouse cursor. At left hand side of the window, the tree structure of the constructed processing plant is shown. *Modelsaz* is a multi-document application and several processing systems can be opened simultaneously.

A two-layered system representation is used for showing physical topology. In each level of the hierarchy, the focused system, called *parent system*, is shown in the form of a large window within the main window. This window is called *parent window* and has a blue title bar showing the name of the parent system and its level. Only immediate sub-systems of the parent system are shown within the parent window by specific icons. For the distinction between simple and composed systems, two different icons are implemented. Other systems in the same level of the parent system are shown outside the parent window by other icons.

Only connections between visible child systems and also between these systems and surrounding of the parent system are shown as arrows with a specific color depending on the connection type. The direction of these arrows is from source to target systems. A combo box is used to store some information about all connections between two systems and is called *connection box*. Any connection between two systems can be selected from this box. A pink color icon is shown at the right hand side of any connection box, which is sensitive to the right click of the mouse and upon it, a pop-up menu will be appeared. This menu has some instructions for changing the properties of a selected connection.

A view of the application consisting of different parts is demonstrated in Fig. 6. At the stage of physical topology construction, systems are similar to empty containers and mass connections are the same as pipes, which nothing pass through them. At this stage, species of the plant must

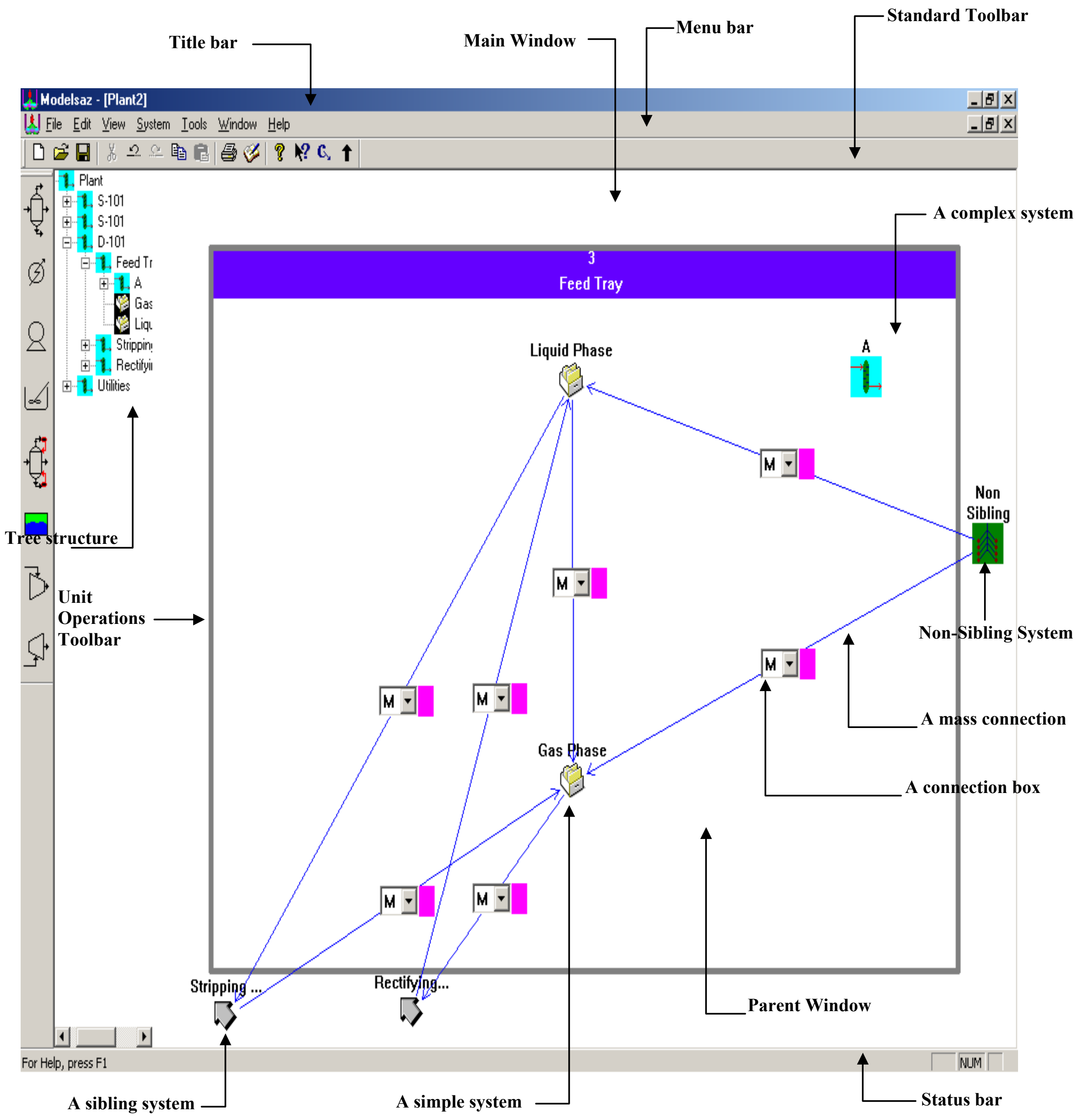


Fig. 6: Demonstration of various parts of Modelsaz graphical interface

be specified. This can be done by *Select Plant Species* command from *Tools* menu. Then the application attempts to find, from a reactions database, all reactions that can occur between species of the plant and adds them to the plant reactions set. The user can activate or deactivate any reaction in this set. After addition of species to the plant, some of these species must be injected to simple systems. By selecting *Inject Species* command from *System* menu a dialog box appears and the user can define the species of a selected simple system from the plant species set. By pressing *OK* button all selected species are injected to the system (Fig. 7). Then, the application will construct species topology of the plant using the *Coloring* and *Add Species* algorithms.

After the construction of species topology, some correlations and properties such as kinetic and transfer laws, physical and thermodynamic properties, and state variable transformations can be specified. All equations specified for a composed system will be inherited automatically by sub-systems of the selected system based on inheritance principle. By using *Generate Report* command from *Tools* menu, one can generate mathematical model of a constructed plant. This command displays a dialog box, which asks the name of a file for saving model equations. By entering the file name and selecting *Save* button, model equations comprising of dynamic mass and energy balance equations, kinetic laws, transfer laws, state variable transformations, and other information are stored in the file. The format of the generated equations is compatible with *Mathematica*™ syntax.

Constructed processing plants can be stored in a

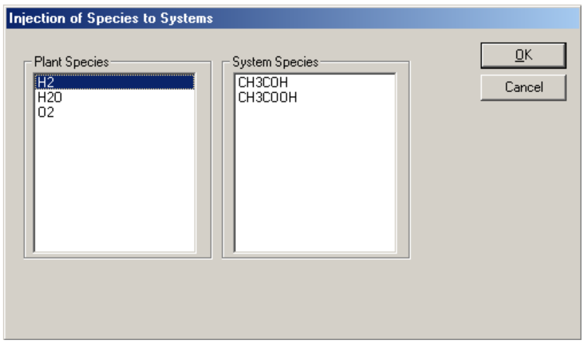


Fig. 7: Injecting some species to a system

model library for next uses. Reusability of generated models is one the main advantages of this software, which avoids users from doing a lot of iterative tasks.

Some of the applied operations such as creation or deleting a system or a connection can be neglected by *Undo* command from *Edit* menu. Furthermore, any neglected operation can be done again by *Redo* command of the same menu. This software package has many utilities for helping users in the construction of process models. Detailed description of the software can be found in the literature [23].

Case Study

In this section, the graphical and mathematical model of hydrodealkylation of toluene (HDA) process is presented. Fig. 8 shows the simplified flowsheet of this process and Fig. 9 represents its physical structure which has been constructed within the software. Some parts of the generated report in *Mathematica*™ syntax are shown below:

```
****************************************************************)
Model Name:     HDA Process
==========================================================================
                         HIERARCHICAL STRUCTURE
==========================================================================
System Name        System Id     Father Id    Number of Childs    Child Systems
-----------------  ------------  -----------  ------------------  ----------------
HDA Process            1             0              9             {2,9,21,57,58,223,224,225,228}
Product              228             1              0
Flash Drum           225             1              2             {226,227}
Gas Phase            227           225              0
Liquid Phase         226           225              0
Split                224             1              0
Split                    223          1          0
Columns                   58          1          3          {59,119,174}
Benzene Column           174         58          1          {175}
Distillation Tower       175        174          3          {176,179,201}
Rectifying Section       201        175          7          {202,205,208,211,214,217,220}
Tray1                    220        201          2          {221,222}
 .
 .
 .
================================================================================
                              CONNECTION(S):
================================================================================
TYPE        INDEX                Source System Id(<)-->Target System Id
-------     ----------           -------------------------------------------------
MASS          1                           224<-->19
MASS          2                            12<-->14
MASS          3                            11<-->15
MASS          4                           223<-->11
MASS          5                            56<-->97
MASS          6                            98<-->56
MASS          7                            39<-->139
MASS          8                           138<-->39
MASS          9                            44<-->188
MASS         10                           187<-->44
 .
 .
 .
.================================================================================
                              SPECIES:
================================================================================
                     INDEX                         NAME
                     ---------                     ---------
                        1                          $C_6H_5CH_3$
                        2                          $H_2$
                        3                          $CH_4$
                        4                          $C_6H_6$

================================================================================
                              REACTIONS:
================================================================================
                    INDEX                         FORMULA
                    ----------               --------------------------------
                       1                     $C_6H_5CH_3 + H_2 \longrightarrow C_6H_6 + CH_4$

***************************************************************************)

(***************************************************************************
                    REACTION  KINETICS
                    ( s = System Id )
```

```
**********************************************************************)
rp[1,s_][t_]:=kr[1,s][t]*cons[1,s][t]*cons[2,s][t]*V[s][t]
(**********************************************************************
                              PROPERTIES OF SPECIES:
**********************************************************************)
ac={-0.24354616*10^5,0.27143024*10^5,0.19250906*10^5,-0.33917267*10^5};
bc={0.51246432*10^3,0.9273762*10^1,0.5212566*10^2,0.47436444*10^3};
cc={-0.27653814,-0.13808066*10^-1,0.11974248*10^-1,-0.30170081};
dc={0.49111164*10^-4,0.76450968*10^-5,-0.1131692*10^-4,0.71301204*10^-4};
ec={0,0,0,0};
fc={0,0,0,0};
ggc={0,0,0,0};

ap={-0.2284786*10^2,-0.5175211*10^4,-0.1843663*10^4,-0.37675546*10^2};
bp={-0.1874186*10^4,0.3803658*10^5,0.620333*10^5,-0.1097523*10^4};
cp={-0.7292144*10^2,0.8204062*10^1,0.5891241*10^2,-0.9353251*10^2};
dp={-0.1083663*10^-1,-0.3030702*10^2,-0.8012375,-0.1404773*10^-1};
ep={0.7485147*10^1,0.1466486*10^4,0.335193*10^3,0.9948222*10^1};
fp={0.7567229*10^-17,0.1273497,0.2485396*10^-8,0.104165*10^-16};
ggp={6,2,4,6};

Tlp={236.68,15.66,90.7,278.7};
Thp={597.17,33.2,190.6,562.1};
Mw={92.141,2.016,16.043,78.114};
Tf={178,14,90.7,278.7};
Tb={383.8,20.4,111.7,353.3};
Tc={591.7,33.2,190.6,562.1};
Pc={0.4113795*10^7,0.129696*10^7,0.4600155*10^7,0.48939975*10^7};
Vc={0.316,0.065,0.099,0.259};
Zc={0.264,0.305,0.288,0.271};
Wc={0.257,-0.22,0.008,0.212};
dHf={0.5003226*10^8,0,-0.74901852*10^8,0.82982376*10^8};
dGf={0.12208709*10^9,0,-0.5086962*10^8,0.12974893*10^9};
ak={0,0,0,0};
bk={0,0,0,0};
ck={0,0,0,0};
dk={0,0,0,0};
visc1={467.33,13.82,114.14,545.64};
visc2={0.96736798*10^2,2.4838213,0.2291233*10^2,0.10791118*10^3};

xx[1 ,s_][t_]:=1-xx[2 ,s][t]-xx[3 ,s][t]-xx[4 ,s][t]
num=4 ;

(**********************************************************************
                              PROPERTIES  OF  REACTIONS:
**********************************************************************)
```

```
k0={0};
are={0};
act={0};
dhr={dHf[[4 ]]+dHf[[3 ]]-(dHf[[1 ]]+dHf[[2 ]])};
rtmax={0};
rtmin={0};
rpmax={0};
rpmin={0};
req={0};

rnum=1 ;

(*********************************************************************
                         CONSTANTS AND PARAMETERS:
*********************************************************************)
R:=8314
kr[r_,s_][t_]:=k0[[r]]*TT[s][t]^are[[r]]*Exp[-act[[r]]/(R*TT[s][t])]
Tref:=0
mu[sp_,T_]:=Exp[visc1[[sp]]*(1/(T*visc2[[sp]]))]
gama[sp_,s_][t_]:=1
eff[s1_,s2_][t_]:=1
vli[sp_,s_][t_]:=R*Tc[[sp]]*Zc[[sp]]^(1+(Sign[1-TT[s][t]/Tc[[sp]]]*(Abs[1-TT[s][t]/Tc[[sp]]])^(2/7)))/Pc[[sp]]
VV[s1_,s2_][t_]:=VT[s1,s2]-VL[s1,s2][t]
VVM[s1_][t_]:=R*TT[s1][t]/PP[s1][t]+BB[s1][t]
Nn[sp_,s1_][t_]:=xx[sp,s1][t]*Nt[s1][t]
kcc[sp1_,sp2_]:=0
Tcc[sp1_,sp2_]:=((Tc[[sp1]]*Tc[[sp2]])^0.5)*(1-kcc[sp1,sp2])
Vcc[sp1_,sp2_]:=((Vc[[sp1]]^(1/3)+Vc[[sp2]]^(1/3))/2)^3
Zcc[sp1_,sp2_]:=(Zc[[sp1]]+Zc[[sp2]])/2
Wcc[sp1_,sp2_]:=(Wc[[sp1]]+Wc[[sp2]])/2
.
.
.
 (*********************************************************************
                              TRANSFER LAWS:
*********************************************************************)
nn[sp_,1 ,19 ,224 ][t_]:=kp[sp,1 ][t]*Acon[1 ]*(PPP[sp,224 ][t]-PPP[sp,19 ][t])
nn[sp_,1 ,224 ,19 ][t_]:=kp[sp,1 ][t]*Acon[1 ]*(PPP[sp,19 ][t]-PPP[sp,224 ][t])
nn[sp_,235 ,116 ,221 ][t_]:=kp[sp,235 ][t]*Acon[235 ]*(PPP[sp,221 ][t]-PPP[sp,116 ][t])
nn[sp_,235 ,221 ,116 ][t_]:=kp[sp,235 ][t]*Acon[235 ]*(PPP[sp,116 ][t]-PPP[sp,221 ][t])
nn[sp_,234 ,97 ,118 ][t_]:=kp[sp,234 ][t]*Acon[234 ]*(PPP[sp,118 ][t]-PPP[sp,97 ][t])
nn[sp_,234 ,118 ,97 ][t_]:=kp[sp,234 ][t]*Acon[234 ]*(PPP[sp,97 ][t]-PPP[sp,118 ][t])
nn[sp_,233 ,118 ,98 ][t_]:=kp[sp,233 ][t]*Acon[233 ]*(PPP[sp,98 ][t]-PPP[sp,118 ][t])
.
.
.
(*********************************************************************
                          EQUILIBRIUM RELATIONS:
*********************************************************************)
```

```
VLM[s1_,s2_][t_]:=xx[1 ,s2][t]*vli[1 ,s2][t]+xx[4 ,s2][t]*vli[4 ,s2][t]
PPP[1 ,20 ][t_]:=xx[1 ,19 ][t]*gama[1 ,19 ][t]*Psat[1 ,TT[19 ][t]]/phib[1 ,20 ][t]
PPP[1 ,19 ][t_]:=xx[1 ,20 ][t]*gama[1 ,20 ][t]*Psat[1 ,TT[20 ][t]]/phib[1 ,19 ][t]
PPP[1 ,15 ][t_]:=xx[1 ,14 ][t]*gama[1 ,14 ][t]*Psat[1 ,TT[14 ][t]]/phib[1 ,15 ][t]
PPP[1 ,14 ][t_]:=xx[1 ,15 ][t]*gama[1 ,15 ][t]*Psat[1 ,TT[15 ][t]]/phib[1 ,14 ][t]
PPP[1 ,12 ][t_]:=xx[1 ,11 ][t]*gama[1 ,11 ][t]*Psat[1 ,TT[11 ][t]]/phib[1 ,12 ][t]
PPP[1 ,11 ][t_]:=xx[1 ,12 ][t]*gama[1 ,12 ][t]*Psat[1 ,TT[12 ][t]]/phib[1 ,11 ][t]
PPP[1 ,8 ][t_]:=xx[1 ,7 ][t]*gama[1 ,7 ][t]*Psat[1 ,TT[7 ][t]]/phib[1 ,8 ][t]
PPP[1 ,7 ][t_]:=xx[1 ,8 ][t]*gama[1 ,8 ][t]*Psat[1 ,TT[8 ][t]]/phib[1 ,7 ][t]
PPP[1 ,5 ][t_]:=xx[1 ,4 ][t]*gama[1 ,4 ][t]*Psat[1 ,TT[4 ][t]]/phib[1 ,5 ][t]
PPP[1 ,4 ][t_]:=xx[1 ,5 ][t]*gama[1 ,5 ][t]*Psat[1 ,TT[5 ][t]]/phib[1 ,4 ][t]
PPP[1 ,228 ][t_]:=xx[1 ,225 ][t]*gama[1 ,225 ][t]*Psat[1 ,TT[225 ][t]]/phib[1 ,228 ][t]
PPP[1 ,227 ][t_]:=xx[1 ,226 ][t]*gama[1 ,226 ][t]*Psat[1 ,TT[226 ][t]]/phib[1 ,227 ][t]
PPP[1 ,226 ][t_]:=xx[1 ,227 ][t]*gama[1 ,227 ][t]*Psat[1 ,TT[227 ][t]]/phib[1 ,226 ][t]
PPP[1 ,224 ][t_]:=xx[1 ,223 ][t]*gama[1 ,223 ][t]*Psat[1 ,TT[223 ][t]]/phib[1 ,224 ][t]
PPP[1 ,223 ][t_]:=xx[1 ,58 ][t]*gama[1 ,58 ][t]*Psat[1 ,TT[58 ][t]]/phib[1 ,223 ][t]

(*********************************************************************
                         OTHER  EQUATIONS:
*********************************************************************)
Nt[4 ][t_]:=VV[4 ,5 ][t]/VVM[4 ][t]
xx[sp_,4 ][t_]:=eff[4 ,5 ][t]*PPP[sp,4 ][t]/PPP[4 ][t]
TT[4 ][t_]:=T1[4 ,5 ][t]
V[4 ][t_]:=VV[4 ,5 ][t]
T1[4 ,5 ][t_]:=298.15
np[2 ,4 ][t_]:=-rp[1 ,4 ][t]
np[4 ,4 ][t_]:=rp[1 ,4 ][t]
np[3 ,4 ][t_]:=rp[1 ,4 ][t]
np[1 ,4 ][t_]:=-rp[1 ,4 ][t]
hr[4 ][t_]:=-hp[1 ,4 ][t]
Nt[5 ][t_]:=VV[5 ,4 ][t]/VVM[5 ][t]
xx[sp_,5 ][t_]:=eff[5 ,4 ][t]*PPP[sp,5 ][t]/PPP[5 ][t]
xx[1 ,5 ][t_]:=0
xx[2 ,5 ][t_]:=0
xx[3 ,5 ][t_]:=0
xx[4 ,5 ][t_]:=0
TT[5 ][t_]:=T1[5 ,4 ][t]
V[5 ][t_]:=VV[5 ,4 ][t]
T1[5 ,4 ][t_]:=298.15
Nt[7 ][t_]:=VV[7 ,8 ][t]/VVM[7 ][t]

(*********************************************************************
                         BALANCE  EQUATIONS:
*********************************************************************)
aaa=NDSolve[{
```

```
Nn[2 ,26 ]'[t]==nn[2 ,21 ,26 ,51 ][t]+nn[2 ,20 ,26 ,226 ][t]+nn[2 ,19 ,26 ,227 ][t]+np[2 ,26 ][t],
Nn[4 ,26 ]'[t]==nn[4 ,21 ,26 ,51 ][t]+nn[4 ,20 ,26 ,226 ][t]+nn[4 ,19 ,26 ,227 ][t]+np[4 ,26 ][t],
Nn[3 ,26 ]'[t]==nn[3 ,21 ,26 ,51 ][t]+nn[3 ,20 ,26 ,226 ][t]+nn[3 ,19 ,26 ,227 ][t]+np[3 ,26 ][t],
Nn[1 ,26 ]'[t]==nn[1 ,21 ,26 ,51 ][t]+nn[1 ,20 ,26 ,226 ][t]+nn[1 ,19 ,26 ,227 ][t]+np[1 ,26 ][t],
Nn[2 ,29 ]'[t]==nn[2 ,28 ,29 ,8 ][t]+nn[2 ,27 ,29 ,46 ][t]+np[2 ,29 ][t],
Nn[4 ,29 ]'[t]==nn[4 ,28 ,29 ,8 ][t]+nn[4 ,27 ,29 ,46 ][t]+np[4 ,29 ][t],
Nn[3 ,29 ]'[t]==nn[3 ,28 ,29 ,8 ][t]+nn[3 ,27 ,29 ,46 ][t]+np[3 ,29 ][t],
Nn[1 ,29 ]'[t]==nn[1 ,28 ,29 ,8 ][t]+nn[1 ,27 ,29 ,46 ][t]+np[1 ,29 ][t],
Nn[2 ,31 ]'[t]==nn[2 ,31 ,31 ,34 ][t]+nn[2 ,29 ,31 ,4 ][t]+np[2 ,31 ][t],
Nn[4 ,31 ]'[t]==nn[4 ,31 ,31 ,34 ][t]+nn[4 ,29 ,31 ,4 ][t]+np[4 ,31 ][t],
Nn[3 ,31 ]'[t]==nn[3 ,31 ,31 ,34 ][t]+nn[3 ,29 ,31 ,4 ][t]+np[3 ,31 ][t],
Nn[1 ,31 ]'[t]==nn[1 ,31 ,31 ,34 ][t]+nn[1 ,29 ,31 ,4 ][t]+np[1 ,31 ][t],
Nn[2 ,34 ]'[t]==nn[2 ,33 ,34 ,49 ][t]+nn[2 ,31 ,34 ,31 ][t]+np[2 ,34 ][t],
Nn[4 ,34 ]'[t]==nn[4 ,33 ,34 ,49 ][t]+nn[4 ,31 ,34 ,31 ][t]+np[4 ,34 ][t],
Nn[3 ,34 ]'[t]==nn[3 ,33 ,34 ,49 ][t]+nn[3 ,31 ,34 ,31 ][t]+np[3 ,34 ][t],
Nn[1 ,34 ]'[t]==nn[1 ,33 ,34 ,49 ][t]+nn[1 ,31 ,34 ,31 ][t]+np[1 ,34 ][t],
Nn[2 ,36 ]'[t]==nn[2 ,26 ,36 ,46 ][t]+nn[2 ,25 ,36 ,41 ][t]+np[2 ,36 ][t],
Nn[4 ,36 ]'[t]==nn[4 ,26 ,36 ,46 ][t]+nn[4 ,25 ,36 ,41 ][t]+np[4 ,36 ][t],
Nn[3 ,36 ]'[t]==nn[3 ,26 ,36 ,46 ][t]+nn[3 ,25 ,36 ,41 ][t]+np[3 ,36 ][t],
Nn[1 ,36 ]'[t]==nn[1 ,26 ,36 ,46 ][t]+nn[1 ,25 ,36 ,41 ][t]+np[1 ,36 ][t],
Nn[2 ,39 ]'[t]==nn[2 ,8 ,39 ,138 ][t]+nn[2 ,7 ,39 ,139 ][t]+np[2 ,39 ][t],
Nn[4 ,39 ]'[t]==nn[4 ,8 ,39 ,138 ][t]+nn[4 ,7 ,39 ,139 ][t]+np[4 ,39 ][t],
Nn[3 ,39 ]'[t]==nn[3 ,8 ,39 ,138 ][t]+nn[3 ,7 ,39 ,139 ][t]+np[3 ,39 ][t],
Nn[1 ,39 ]'[t]==nn[1 ,8 ,39 ,138 ][t]+nn[1 ,7 ,39 ,139 ][t]+np[1 ,39 ][t],
Nn[2 ,41 ]'[t]==nn[2 ,25 ,41 ,36 ][t]+nn[2 ,24 ,41 ,54 ][t]+np[2 ,41 ][t],
Nn[4 ,41 ]'[t]==nn[4 ,25 ,41 ,36 ][t]+nn[4 ,24 ,41 ,54 ][t]+np[4 ,41 ][t],
Nn[3 ,41 ]'[t]==nn[3 ,25 ,41 ,36 ][t]+nn[3 ,24 ,41 ,54 ][t]+np[3 ,41 ][t],
Nn[1 ,41 ]'[t]==nn[1 ,25 ,41 ,36 ][t]+nn[1 ,24 ,41 ,54 ][t]+np[1 ,41 ][t],
Nn[2 ,44 ]'[t]==nn[2 ,10 ,44 ,187 ][t]+nn[2 ,9 ,44 ,188 ][t]+np[2 ,44 ][t],
Nn[4 ,44 ]'[t]==nn[4 ,10 ,44 ,187 ][t]+nn[4 ,9 ,44 ,188 ][t]+np[4 ,44 ][t],
Nn[3 ,44 ]'[t]==nn[3 ,10 ,44 ,187 ][t]+nn[3 ,9 ,44 ,188 ][t]+np[3 ,44 ][t],
Nn[1 ,44 ]'[t]==nn[1 ,10 ,44 ,187 ][t]+nn[1 ,9 ,44 ,188 ][t]+np[1 ,44 ][t],
Nn[2 ,46 ]'[t]==nn[2 ,27 ,46 ,29 ][t]+nn[2 ,26 ,46 ,36 ][t]+np[2 ,46 ][t],
Nn[4 ,46 ]'[t]==nn[4 ,27 ,46 ,29 ][t]+nn[4 ,26 ,46 ,36 ][t]+np[4 ,46 ][t],
Nn[3 ,46 ]'[t]==nn[3 ,27 ,46 ,29 ][t]+nn[3 ,26 ,46 ,36 ][t]+np[3 ,46 ][t],
Nn[1 ,46 ]'[t]==nn[1 ,27 ,46 ,29 ][t]+nn[1 ,26 ,46 ,36 ][t]+np[1 ,46 ][t],
Nn[2 ,49 ]'[t]==nn[2 ,35 ,49 ,14 ][t]+nn[2 ,34 ,49 ,15 ][t]+nn[2 ,33 ,49 ,34 ][t]+np[2 ,49 ][t],
Nn[4 ,49 ]'[t]==nn[4 ,35 ,49 ,14 ][t]+nn[4 ,34 ,49 ,15 ][t]+nn[4 ,33 ,49 ,34 ][t]+np[4 ,49 ][t],
.
.
.
},{
},{t,0,2000}
,MaxSteps->1000000
]
(*
Cell[Null,InitializationCell]
*)
```

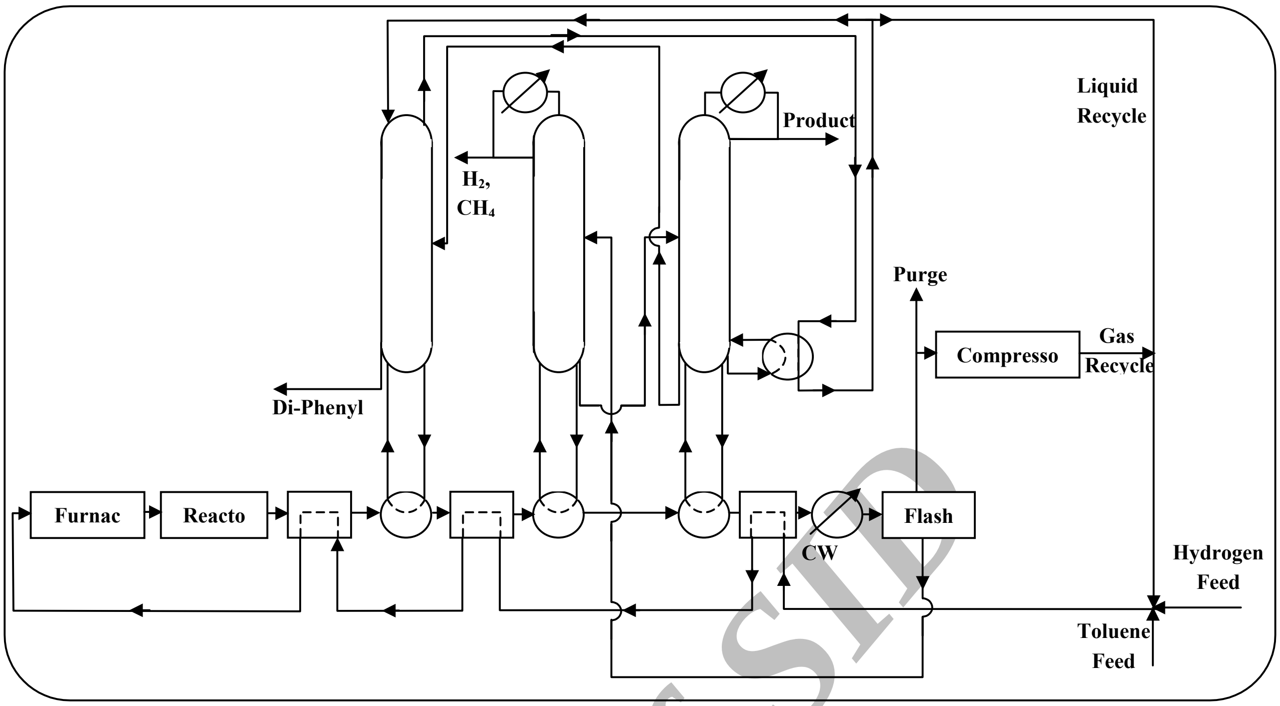


Fig. 8: Simplified Flowsheet of Hydrodealkylation of Toluene (HAD Process)

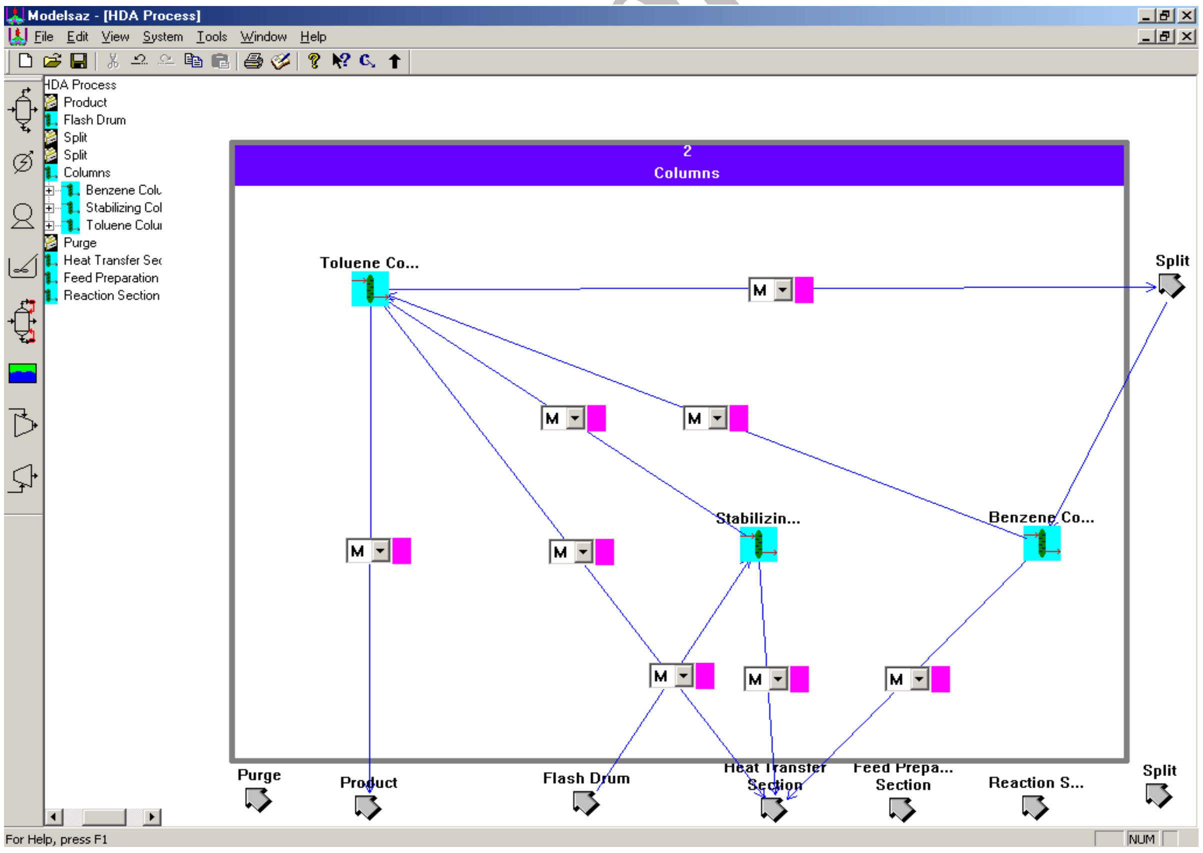


Fig. 9: HAD process Structure, Created in Modelsaz

CONCLUSION

In recent years lots of works have been done for the development of advanced modeling and simulation tools. This paper introduces a new tool, called *Modelsaz,* for the modeling of processing plants in such a way that a user does not need to perform any computer programming. Object-oriented concepts have been widely used in its design and development. This application meets necessary requirements for modeling such as complexity management, storing, retrieving and reusing of the models, etc.

This software is under development in order to support modeling of distributed-parameter systems, manipulation of algebraic equations, simulation of processing plants [23] and integrating with various solvers.

Received: 15th October 2002 ; Accepted: 25th July 2003

REFERENCES

[1] Wolfram, S., *Mathematica*, University of Cambridge/ Wolfram Media Inc., New York, USA, (1999).

[2] Marquardt, W., Dynamic Process Simulation – Recent Trends and Future Challenges, *Chemical Process Control CPC-IV*, CACHE, Austin, *AIChE*, New York, pp. 131-180, (1991).

[3] Boston, J. F., Britt, H. I. And Tayyabkhan, M. T., Software: Tackling Tougher Tasks, *Chem. Eng. Progr.*, Vol. Nov., pp. 38-49, (1993).

[4] Bhargava, H.K, "Formal Semantics of a Typed Modeling Language, ASCEND, Smeal College of Business Administration", Penn State University, http://www.smeal.psu.edu/~bhargava/, 41 pages, Octobor (2001).

[5] Elmqvist, H. E., Brück, D. and Otter, M., *Dymola – User's Manual*, Dynasim AB, Lund, Sweden, (1996).

[6] Lund, P. C., *An Object-Oriented Environment for Process Modeling and Simulation*, Ph.D. Thesis, Laboratory of Chemical Engineering, Norwegian Institute of Technology, Trondheim, (1992).

[7] Barton, P. I., The Equation Oriented Strategy for Process Flowsheeting, Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139, 24 pages, March (2000).

[8] Woods, E.A., The Hybrid Phenomena Theory, Ph.D. Thesis, Division of Engineering Cybernetics, Norwegian Institute of Technology, Trondheim, Norway, (1993).

[9] Sørlie, C. F., A Computer Environment for Process Modeling, Ph.D. Thesis, University of Trondheim, Laboratory of Chemical Engineering, Trondheim, Norway, (1990).

[10] Stephanopoulos, G., Henning, G. and Leone, H., MODEL.LA A Modeling Language for Process Engineering. Part I and II. The Formal Framework, *Computers and Chemical Engineering*, **14**(8), pp. 813-869, (1990).

[11] Mehrabani, A. Z., Computer Aided Modelling of Physical-Chemical-Biological Systems, Ph.D. Thesis, University of New South Wales, Australia, (1995).

[12] Asbjørbsen, O. A., Control and Operability of Process Plants; *Computers and Chemical Engineering*, Vol. **13**(4,5), pp. 34-42, (1994).

[13] Anderson, M., Object-Oriented Modeling and Simulation of Hybrid Systems, Ph.D. Thesis, Dep. of Automatic Control, Lund Institute of Technology, Lund, Sweden, (1994).

[14] Telnes, K., Computer-Aided Modeling of Dynamic Processes Based on Elementary Physics, Ph.D. Thesis, Division of Engineering Cybernetics, Norwegian Institute of Technology, Trondheim, (1992).

[15] Krobb, C., Lohmann, B. and Marquardt, W., *The* Chemical Engineering Data Model*,* VeDa. Part 6: The Process of Model Development, Internal Report, Lehr-und Forschungsgebiet Theoretische Informatik, RWTH Aachen, (1998).

[16] Bär, M. and Zeitz, M., A Knowledge-based Flowsheet-oriented User Interface for a Dynamic Process Simulator, *Comp. Chem. Eng.*, **14**, pp. 1275-1283, (1990).

[17] Marquardt, W., Trends in Computer-Aided Process Modeling, *Comp. Chem. Eng.*, **20**(6,7), pp. 591-609, (1996).

[18] Andersson, M., Object Oriented Modeling and Simulation of Hybrid Systems, Ph.D. Thesis, Department of Automatic Control, Lund Institute of Technology, Sweden, (1994).

[19] Farzi A., Modeling Physical-Chemical-Biological Systems by the Aid of the Computer, M.Sc. Thesis,

Isfahan University of Technology, Dep. of Chemical Engineering, Isfahan, Iran, 2000 (in Farsi).

[20] Mehrabani, Z. A. and Farzi, A., Mathematical Representation of Tree Structures for Processing Systems, *Iran. J. Chem. & Chem. Eng.*, Tehran, Iran, (2002).

[21] Veverka, V.V. and Modran, F., Material and Energy Balancing in the Process Industries, From Microscopic Balances to Large Plants, *Elsevier,* Amsterdam, (1997).

[22] Williams, M., Teach Yourself Visual C++6 in 24 Hours, SAMS, (1998).

[23] Farzi, A., A. Mehrabani Z. and Etemad, S. Gh., SimuChemPro: An Object-Oriented Environment for Modeling and Simulation, Proceedings of Chisa 2002 Congress, Czech, Praha, G2.6, (2002).