Carnegie Mellon University

Department of Chemical Engineering

Modeling the Physical Behavior and the Operation of Chemical Processes in a Purely Declarative Modeling Environment

Written statement submitted as a requirement for the Ph. D. qualifying examination by:

Vicente Rico-Ramirez

Advisor: **A.W. Westerberg**

Pittsburgh, PA., August 18, 1995.

Modeling the Physical Behavior and the Operation of Chemical Processes in a Purely Declarative Modeling Environment

by Vicente Rico-Ramirez

Most current dynamic simulation packages are mainly concerned with the description of continuous processes. However, the dynamic behavior of processing systems exhibits both continuous and discrete aspects. Recently, the combined discrete-continuous dynamic simulation problem has been studied. These studies propose the decomposition of the processing system into two subsystems: the physical behavior, which is modeled in a declarative language, and the operation of the plant, which is modeled in a procedural language. Because procedural information is included into the modeling representation, the range of activities for which this approach can be used is limited. There is a need for a purely declarative modeling environment to encompass a number of different activities, such as steady state simulation and design, dynamic simulation, steady state optimization, etc. The ASCEND modeling language provides a complete representation of a model in a purely declarative language. Nevertheless, its capabilities for modeling discrete event simulation problems is limited. In this paper, we propose an extension to its modeling capabilities, such that one is able to model both the physical behavior and the operation of the plant by using only a declarative language. This represents a new approach in the field of equation based modeling. The extensions proposed are: 1) the use of conditional modeling, 2) incorporation of logical equations as part of the model representation, and 3) the addition of field variables to the modeling language. Examples are presented to show how the proposed extensions can be used for modeling the operation of a process. The problems involved in the implementation of these ideas are discussed. This paper is primarily concerned with modeling issues, but it is clear that the main effort of future research will be focused in the solution of computational and mathematical problems arising in the combined continuous-discrete dynamic simulation and optimization of chemical processes.

1. Introduction

Dynamic simulation is becoming a common tool for plant design and operation. Potential applications of such a tool have been advocated for many years: safety and risk analysis, operator training, etc.

Most of the current dynamic simulation packages were developed to address the solution of models of predominantly continuous in nature (Marquardt, 1991). However, chemical processes are not strictly continuous. They can change their structure discontinuously at points in state space. Common examples cited are phase changes and flow transitions. Besides these process inherent discontinuities, the majority of processes also exhibit discrete manipulations as a result of the application of operational strategies. Examples include valving operations, regulatory control actions, and startup procedures. Furthermore, the situations in which these discrete changes arise often correspond to those in which it is most worthwhile to perform dynamic simulation.

One of the more recent approaches to the discrete event simulation problem has been described in the development of the simulation package gPROMS. In that work, Barton (1992) concluded that the analysis of the time dependent behavior of a chemical process requires the system to be decomposed into the continuous-discrete physical behavior of the plant (models) and the external actions imposed on it by its environment (tasks). He argues that, in order to promote the reusability of a model, the equations describing the system should be decoupled from the tasks that one wishes perform with the equipment in that process. Because of this decomposition, the physical behavior is specified in a purely declarative language whereas the operation becomes part of the procedural knowledge required to solve the equations.

In this work, a new approach to the discrete event simulation problem is introduced. The goal of this paper is to show that the ASCEND (Piela, 1989; Westerberg *et al*, 1994) modeling language, while keeping the essence of a highly structured declarative language, is rich enough to model not only the physico-chemical behavior of unit operations, but also the operation of the plant (tasks and processes). In order to deal with discrete event simulation problems, we are proposing: 1) the use of conditional modeling, 2) the incorporation of logical equations as part of the model representation, and 3) the addition of field variables to the modeling language. It is important to emphasize that we intend to maintain the equational based approach; that is, the modelers have to state the equations that must be true at the solution but not how to solve these equations. Thus, the reusability of the models is still enhanced. Moreover, the expressiveness and the scope of the problems which one is able to model increases considerably with the proposed extensions. This paper is primarily concerned with modeling issues, but the advantages and problems involved in the implementation of these ideas will also be discussed.

In the next section, a brief review of relevant work done in the field of dynamic simulation is presented. Section 3 contains an analysis of the capabilities and limitations of the current version of the ASCEND modeling language. In addition, examples are presented to show how the extensions proposed above can be used to model the operation of chemical processes. Asymmetric transitions are also analyzed. Finally, in Section 4 conclusions are drawn and ideas for future work are suggested.

2. Previous Work

Many approaches to the dynamic simulation problem appear in the literature. An exhaustive list of the software packages currently available and their general characteristics can be found in Marquardt (1991); Barton (1992) discusses the inadequacies and advantages of each of them. Pantelides and Barton (1994) also present a brief review of the status and future perspectives of the equation-oriented dynamic simulation.

Besides the dynamic simulation packages developed to address the solution of models of predominantly continuous nature, some designs of special purpose dynamic simulation packages for batch processes such as BATCHES (Joglekar and Reklaitis, 1984), have also been reported. Neither of them is suitable for the detailed analysis of systems where both continuous and discrete changes take place and interact to a significant extent during part or all of the period under investigation (Barton, 1992). For instance, SpeedUp (Pantelides, 1988) provides great flexibility when constructing the dynamic models of an entire process. However, according to Pantelides and Barton (1994), it provides limited capabilities for the description of the dynamic simulation problems in which discrete changes and discontinuities significantly affect the overall behavior; its IF construct only can be used for the declaration of reversible discontinuities. Furthermore, the only form of structural manipulation tolerated by SpeedUp is the replacement of a set of input equations for a new set of input equations involving the same number of variables.

Any kind of modeling discontinuity can be represented in the model description of gPROMS (Barton, 1992). This software package addresses the solution of a combined discrete-continuous dynamic simulation problem. In that work, the physical behavior of a system is modeled in a declarative language whereas the operation of the plant is represented in a procedural language. Hence, while simulation clock advances, the schedule of tasks is executed and any event is detected and processed. The main disadvantage of such an approach is that the range of activities for which it can be used is limited; it is suitable for the modeling and solution of an initial value problem, but it does not provide facilities for optimization of neither steady state nor dynamic nature. Some other deficiencies are identified: *i*) it does not provide an appropriate user interface; the complexity of any activity involving dynamic simulation makes the design of the interface an issue of key importance, and *ii*) in the case of problems involving physico-chemical discontinuities, its algorithm for initialization does not address the issue of simultaneously determining the set of equations which holds and the consistent values for the variables.

There is a critical need for a purely declarative modeling environment which allows a model to be used for a number of different activities, such as steady state simulation and design, dynamic simulation, steady state and dynamic optimization, etc. In summary, there has been a significant progress in dynamic process simulation, but a lot of work still remains to be done.

3. Dynamic Simulation in the ASCEND Modeling Environment

A novel approach to the discrete event simulation problem is introduced in this section. The ASCEND (Advanced System for Computations in ENgineering Design) modeling language is based on object oriented concepts and provides a complete representation of a model in a purely declarative language. An extension to its modeling capabilities will be proposed, such that the operation of the plant can also be modeled in a declarative unifying framework. Since a declarative model representations is independent from any solution algorithm to be used, the range of activities for which a model can subsequently be employed is broadened.

3.1 Current Status of the ASCEND System

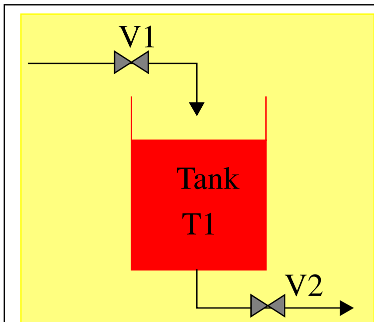
Westerberg *et al.* (1994) discuss the essential features of the current ASCEND system and present several ways on which we can improve ASCEND to solve larger problems and to increase the kinds of processes we can model. This software package provides a symbolic and highly structured declarative representation of models. Reusability, as the main advantage of the equation oriented approach, is enhanced. Practice with this package involves the solution of a wide variety of simulation and optimization problems. Besides the modeling language, the user interface provides facilities for solving and debugging the models. However, the ASCEND capabilities for dealing with discrete events are fairly limited. Even though dynamic simulation problems have been solved, this has been efficiently done only with continuous processes in nature.

Physico-chemical discontinuities can be represented by using conditional modeling (Zaher, 1991). Conditional models exist where the defining equations differ depending on where the solution of the model lies. The syntax for incorporating conditional dependence of some of the equations in the model with minimal ambiguity was suggested by Piela (1989). The work of Zaher

(1995), which adds the CASE-WHEN statement to the ASCEND language and introduces the boundary crossing algorithm, is about to be incorporated. As it will be implemented, the solver will have to select the region in which to find the solution as well as to solve the equations in this region. Initial guesses can be in another region, and the solver has to be able to move from the initial region to one where it can find a solution. However, even the CASE-WHEN statement is not sufficient by itself for modeling irreversible or asymmetric transitions. In addition, ASCEND can only solve and optimize problems where the model contains variables which can be represented only as real numbers. Hence, perhaps the first way to broaden the scope is to allow a modeler the ability to use more types of variables: integer, logical, stochastic and field variables.

3.2 Proposed Extensions

In this section, we propose an extension of the ASCEND modeling language for dealing with discrete event simulation. We intend to keep the equation oriented approach and still be able to model not only the physical behavior of a system but also the operation of the plant. In a declarative language such as ASCEND, the first thing we have to worry about is to get the correct representation for our problem. The system shown in the Fig.1 will be used to illustrate how one can write an ASCEND model for dynamic simulation of a process.



Operational conditions:

1) V1 and V2 can not have the same status at the the same time. If V1 is open, V2 is closed and conversely.

2) If at any time V1 is closed, it will remain closed until a time in which the tank is empty.

3) If at any time V1 is open, it remains open until the tank is filled

Fig. 1 System formed by a tank and two valves

3.2.1 Use of Conditional Modeling

The ASCEND model for this simple flowsheet is presented in the Fig. 2. Hierarchical submodel decomposition is applied such that the flowsheet is constructed by using instances of the

```
MODEL valve;
      open                          IS_A        boolean;            (* open can be TRUE or FALSE *)
      input,output                  IS_A        stream;
      CASE open WHEN                            (*  CASE-WHEN statement *)
            F:  output.flow = 0.0;
            T:  output.flow = function of ΔP and valve parameters (example, K*ΔP);
      END case;
END valve;
MODEL tank;
      input,output                  IS_A        stream;
      level                         IS_A        distance;
      cross_section                 IS_A        area;
      empty, filled                 IS_A        boolean;
      cross_section : = 1.0 { m^2 }
      level$ = (input.flow - output.flow)/cross_section;      (* The symbol $ indicates derivative *)
      filled = (level > = 9 {m});
      empty = (level < = 2 {m});
END tank;
MODEL flowsheet;
      V1,V2                         IS_A        valve;
      T1                            IS_A        tank;
      V1.output, T1.input           ARE_THE_SAME;
      T1.output, V2.input           ARE_THE_SAME;
END flowsheet;
```

Fig. 2 ASCEND model for a system formed by a tank and two valves

models valve and tank. Conditional modeling (CASE-WHEN statement) is used to represent the behavior of the output flow of a valve depending on its status (open or close). In addition, the status of the valve is represented by using the boolean variable open; its value can be TRUE (T, valve is open) or FALSE (F, valve is close). Previous applications of conditional modeling were focused in process inherent discontinuities; here it will be used for modeling the discrete changes occurring because of operational conditions (close or open a valve during a dynamic simulation).

3.2.2. Addition of Field Variables to the Modeling Language

To represent the dynamic behavior of the system, the model Transient_System is created (See Fig.3). The interval over which the simulation is performed is defined with the variable time_interval. In addition, the concept of field variable is introduced: we define an infinite set of instances of our model such that each variable is considered to be available at any point in time over the interval previously defined. The inclusion of field variables provides a very convenient

```
MODEL Transient_System;
      time_interval              IS_A          interval  OF      time;
      TS[time_interval]          IS_A          flowsheet; (*infinite set:  each field variable available at any t*)
      ε                          IS_A          time;
      TS[time_interval].V1.valveparameters                        ARE_THE_SAME;      (*  K  *)
      TS[time_interval].V2.valveparameters                        ARE_THE_SAME;      (*  K  *)
      TS[time_interval].T1.tankparameters                         ARE_THE_SAME;      (*cross_section*)
      FOR t IN time_interval CREATE                              (*logical equations: *)
        TS[t].V1.open = ( NOT( TS[t-ε].T1.filled ) AND ( TS[t-ε].V1.open ) ) OR(TS[t-ε].T1.empty);
        TS[t].V2.open = NOT (TS[t].V1.open);
      END;
END Transient_System;
```

Fig. 3 Dynamic model for the system shown in the Fig. 1

representation of the time dependent behavior of chemical processes, including any discrete change. The compiling implications of the use of these variables are very interesting. Field variables can be thought to be defined over an infinite set. ASCEND already supports allowing one to define anything over a finite set. For a finite set, we require the sets to be explicitly defined before the compiling commences. Thus, the compiled data structure contains storage space for every variable and equation defined over such a set. On the other hand, for an infinite set the compiler cannot set aside the storage space as the solver will control how many points in time it needs as it solves. What can be compiled, however, is one instance in time of the models. The solver can use the compiled instance to determine the time derivatives and the algebraic variables, given values for the states. Thus, the solver can use this same model repeatedly for different values of the states to integrate forward in time.

3.2.3 Incorporation of Logical Equations and Logical Operators

The operational conditions for our model can be represented as a group of logical equations which must be satisfied at every point in the time interval. See Fig. 3. Since we are including boolean variables as unknowns and logical equations as part of our models, we suggest the incorporation of operators which simplify the representation of logical conditions existing among these variables. These operators are the logic *if-then* and *iff-then*.

Regardless the application that we are given to the logical equations, their inclusion represents a new concept in the field of equation based modeling. In fact, the could be used for steady state problems in which logical constraints are imposed. The incorporation of logical equations to the modeling language affects the compilation and solution of the models in ASCEND in the following manner. Boolean variables already can be used as flags, but we have to alter the compiler to allow logical equations and the logical operators to be a part of the problem. In spite of this, this task is not really the main problem. The principal effort of adding boolean variables as unknowns is to have the solvers that can deal with them. Each logical equation will have a logical residual attached to it. This residual will indicate if the expression is satisfied or not. If it is not, we envision solvers that will iterate until such equations are all satisfied. Thus, the essence of an equation oriented approach is kept. Raman and Grossmann (1994) have developed an algorithm for solving logic based integer programming problems. They use formal procedures for performing symbolic inference depending on the way in which the logical equations are represented. Examples include the solution of linear constrained problems using a branch and bound based technique. In addition, recent algorithms have been proposed for solving discrete event dynamic simulation problems (Barton, 1992; Gopal and Biegler, 1994). A solver in which a technique for solving the DAE system and the procedures for performing logical inference can be integrated will have to be provided.

3.2.4 Some Final Comments

A large number of examples of discrete event simulation have been modeled by simultaneously using the extensions proposed above. The CASE-WHEN statement, in combination with the use of logical equations, offers a convenient way to represent complex operational constraints. Irreversible and asymmetric transitions (Fig. 4 and 5) and even advanced modeling tools such as the Petri nets (Yamalidou and Kantor, 1991) can also be successfully represented. In the examples shown (See Fig. 3 and 5), we used an epsilon “ε” to represent a small period of time. Its use can seem ambiguous; however, in most of the real chemical processes, this "ε" usually has a physical interpretation. It can represent the time spent opening a valve, for example Another option would be to use [t-ε] just as a way to represent a point "immediately" before in time. In such a case, this expression is equivalent to the OLD command of gPROMS or to the mathematical representation t^-.

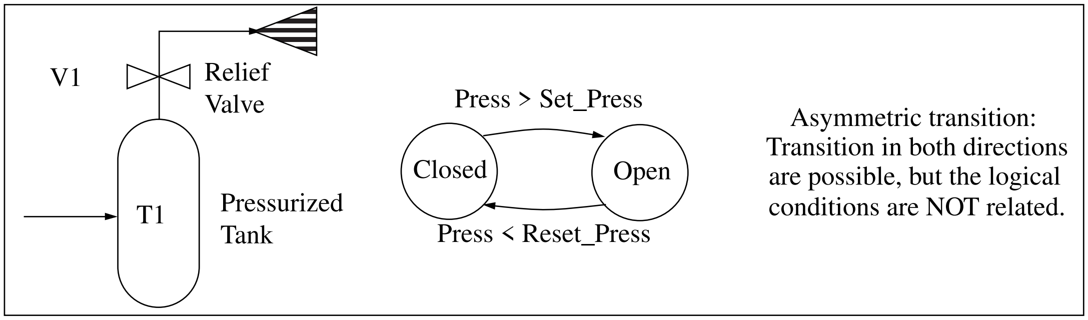


Fig.4 Asymmetric transition: Vessel fitted with a safety relief valve (Barton and Pantelides, 1994)

```
FOR t IN time_interval CREATE
    CASE  RS[t-ε].V1.open  WHEN
       T:  RS[t].V1.open = ( RS[t-ε].T1.Pressure > = 8 {atm} );      (* Reset_Press = 8 {atm} *)
        F:  RS[t].V1.open = ( RS[t-ε].T1.Pressure > = 10 {atm} );     (* Set_Press = 10{atm}  *)
    END case;
 END;
```

Fig.5 Conditional logical equations for the model of an asymmetric transition

4. Conclusions and Future Work

We have used the ASCEND constructs that exist or that we have proposed in this paper to model both the description of a process and the tasks that one wishes perform using the equipment in that process. In other words, we have modeled both the physical behavior and the operation of the plant by using only a declarative language. This approach offers the advantage of broadening the range of activities for which a model can be employed. The features of the ASCEND system allow the modeler to establish a hierarchy of model at any level; therefore, contrary to the arguments of previously reported approaches, reusability is still enhanced. In summary, the expressiveness and the scope of the modeling language have been considerably increased with the incorporation of field variables, logical equations and the use of conditional modeling.

Basically, three steps in our effort to extend the ASCEND capabilities can be identified:

1)defining the syntax for the new tools incorporated to the modeling language, 2) modifying the ASCEND compiler such that it can identify these new tools and create the corresponding data structures, and 3) providing ASCEND with an appropriate solver for dealing with discrete event simulation and logic inference. As discussed earlier, we have accomplished just the first step. Clearly, future work will include any activity involved in the completion of each of the two remaining steps. In fact, current work is focused in learning and managing the compiler.

5. References

Barton, P.I.; The Modeling and Simulation of Combined Discrete/Continuous Processes. PhD Thesis. Department of Chemical Engineering. Imperial College, London, 1992.

Barton, P.I.; Pantelides, C.C.; Modeling of Combined Discrete/Continuous Processes. AIChE J., Vol.40, No. 6, pp. 966-979, 1994.

Gopal, V.; Biegler, L.T.; Nonsmooth Dynamic Simulation with Linear Programming Based Methods. Paper 220g, AIChE Annual Meeting, San Francisco, CA, 1994.

Joglekar, G.S.; Reklaitis, G.V.; A Simulator for Batch and Semi-continuous Processes. Comput. Chem. Engng., Vol. 8, No. 6, pp 315-327, 1984.

Marquardt, W; Dynamic Process Simulation -Recent Progress and Future Challenges. In Chemical Process Control, pp 131-180. CACHE-AIChE Publications, 1991.

Pantelides, C.C.; SPEED-UP -Recent Advances in Process Simulation. Comput. Chem. Engng. Vol. 12, No. 7, pp. 745-755, 1988.

Pantelides, C.C.; Barton, P.I.; Equation-Oriented Dynamic Simulation Current Status and Future Perspectives. Comput. Chem. Engng., Vol. 17, Suppl., pp. S263-S284,1994

Piela, P.C.; ASCEND: An Object Oriented Computer Environment for Modeling and Analysis. PhD Thesis, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA, 1989.

Raman, R.; Grossmann, I.E.; Modeling and Computational Techniques for Logic Based Integer Programming. Comput. Chem. Engng., Vol. 18, No. 7, pp. 563-578, 1994.

Westerberg, A.W.; Abbott, K.; Allan, B.; Plans for ASCEND IV: Our Next Generation Equational-Based Modeling Environment. Presented at Aspen World ‘94, Boston, Massachusetts, November1994.

Yamalidou, E.C.; Cantor, J.C.; Modeling and Optimal Control of Discrete-Event Chemical Processes Using Petri Nets. Comput. Chem. Engng., Vol. 15, No. 7, pp. 503-519, 1991.

Zaher, J.J.; Conditional Modeling. PhD Thesis. Department of Chemical Engineering. Carnegie Mellon University, Pittsburgh, PA, 1995.

Zaher, J.J.; Conditional Modeling in an Equation-Based Modeling Environment. Paper 138c, AIChE Annual Meeting, Los Angeles, CA, 1991.