

# A Virtual Plant Modeler (VPMOD) for Batch Chemical Processes

Chell A. Roberts, Arizona State University, Tempe, AZ  
Maged M. Dessouky, University of Southern California, Los Angeles, CA  
Yasser M. Dessouky, Miami University, Oxford, OH.

## Abstract

This paper presents a Virtual Plant Modeler, VPMOD, which formally characterizes and integrates chemical product designs, batch-chemical equipment (plants), the real-time scheduling of chemical batches, and the control of the chemical transport through the plant. These models provide a framework for agile batch chemical manufacturing that has the ability to automatically reroute and control chemical product flow in a flexible plant subject to unexpected events, such as changes in demand patterns and equipment failure. A formal logic model is generated to control the actual system events, which are non-deterministic. A simulation environment in VPMOD is used to validate schedules and control logic based on plant models supplied by industry. The formal models have been implemented in an object-oriented language.

Keywords: Virtual Factory, Real-time Control, Batch Chemical Manufacturing, Object-oriented

## 1. Introduction

Manufacturing of chemical products has traditionally followed a continuous process dedicated to one or a few products, with each product processed one at a time over a long duration. In the late 1970's and early 1980's the market pressure for greater product variety forced a gradual shift from continuous manufacturing to batch manufacturing. Batch chemical manufacturing is characterized by the production of discrete quantities of chemicals on multipurpose equipment. The batch process approach to chemical manufacturing is effective at economically elevating product mix and responding to varying product quantities. However, efficient production of a high product mix that can optionally be produced on different equipment requires flexible intelligent tools throughout the manufacturing life-cycle embedded in an integrated architecture. Traditional scheduling, planning and integration tools for the continuous process industries do not have this flexibility (Musier and Evans, 1990).

More recently, the intense competition in the market place has created a surge of interest among leading manufacturers and researchers in "agile manufacturing", which provides for further production flexibility and the ability to respond quickly to shifts in demand patterns and unexpected events affecting plant-wide operations such as equipment breakdowns. The next expected evolutionary approach to chemical manufacturing is agile manufacturing (Hofman, 1993). It is anticipated that this approach will increase productivity by improving equipment utilization, reduce waste by providing alternative routes to products which would be subject to spoilage due to equipment breakdown, and increase competitiveness by providing the plant with the ability to adapt readily to customer demand. This next evolution necessitates an extension to existing theory and the development of new techniques coupled with specialized computer aided tools.

Most of the recent advances in software engineering and design of complex systems have not been exploited in the process industries (McCarthy, 1990, Suydam, 1987). These advances have the potential of solving many of the present problems. Among these tools are high level simulation tools (Baudel et. al, 1988) that can be combined with expert systems, knowledge based tools, and object oriented techniques to form an integrated, high level software solution to the flexibility issue.

A set of integrated models in an interactive object oriented environment was developed in response to these needs. These models, hereafter called VPMOD (Virtual Plant Modeler),

formally characterizes and integrates chemical product designs, batch-chemical equipment (plants), the real-time scheduling of these chemicals, and the control of the chemical transport through the plant. VPMOD was developed to exploit the natural flexibility of the batch chemical manufacturing process (i.e., alternative equipments) and to operate in a dynamic, on-line environment.

VPMOD is capable of responding to many real-time system events including the arrival of new orders, the modification of existing orders including quantity or priority changes, equipment failure, and stochastic processing times. Traditional scheduling systems typically assume process starting and ending times to be deterministic. Schedules are determined and act as a guide to the actual process. In reality, processing times are stochastic. Changes in the schedule, such as re-routing, are complicated by in-process inventory. VPMOD has the capability of reactive, interactive scheduling that includes re-routing while considering in-process inventory. VPMOD generates control logic corresponding to the schedule for chemical transport.

VPMOD utilizes a simulation mode to demonstrate and validate these capabilities. The user of VPMOD can interactively generate these events or they can be incorporated into the simulation architecture. The plant model is then updated on-line, in real-time with these events, which are passed to the scheduler. The scheduling system then determines the appropriate action to take based on specified rules. The user has the ability to over-ride the changes and implement or test other possibilities. The scheduling decision is passed to the logic control generator, which implements the decision into a formal control logic model.

## 2. Batch Chemical Manufacturing

Batch chemical plants are classified as either multiproduct or multipurpose (Rippin, 1983). Multiproduct batch chemical plants are analogous to flow shops in discrete manufacturing where all products follow the same sequence of operations (a single process path). Multipurpose batch chemical plants are analogous to job shops with alternative routing in discrete manufacturing where products may follow different process paths. The focus of this paper is on multipurpose batch chemical plants. A configuration of a multipurpose batch chemical process plant is shown in Figure 1. This configuration is a hybrid of actual plants produced by Honeywell for experimentation. The configuration allows the manufacture of multiple chemical consumer goods using the same equipment resources. The main objective is to blend multiple ingredients

from a set of input tanks to produce an end product. In this example there are four input tanks coupled through one header (junction) to four mixers. The mixers are coupled to four reactors and the reactors are coupled to output storage tanks.

(\*\*\*\*\* Insert Figure 1 here \*\*\*\*\*)

There are several unique features of batch chemical manufacturing. First, it is possible and often desirable to split or merge batches to make effective use of equipment resources. This provides flexibility in processing orders of various sizes, since larger orders may be divided into smaller batches, and smaller orders of the same product may be combined into larger batches. Second, the batch processing time is typically independent of the batch size, accordingly estimates of processing times can be made before batch sizes are determined. Third, there are many alternative operations that can manufacture a particular chemical type, and a specific operation can be processed by alternative equipment sets. Routing these batches through batteries of multipurpose machines allows the production of greater varieties of product mixes in different quantities, meeting the market needs for greater product variety, but adds to the complexity of scheduling production.

### 3. Related Work

#### 3.1 Plant Models

Several authors discuss the need to develop virtual manufacturing environments in order to achieve agile manufacturing (McGehee et al., 1994, Parks et al., 1994, Witzerman and Nof, 1995, Cho et al., 1996). Virtual manufacturing environments can assist the company in its efforts to rapidly and effectively react to changes in market conditions and technology.

McGehee, Hebley, and Mahaffey (1994) present a framework, Microelectronics Manufacturing Science and Technology (MMST), for the semiconductor industry to support the transition from high volume production to low volume part specialization. MMST is based on an open distributed system and object-oriented technologies. The major applications of MMST are: factory manager, factory planner, factory scheduler, factory simulator, specification manager, generic equipment model, machine control, and process control. MMST is implemented in Smalltalk.

Parks et al.(1994) propose an Integrated Manufacturing Design Environment (IMDE) which unifies the various design tools and models to support concurrency in the design life cycle. The major components of IMDE are: data model which represents the entities of the system, intelligent interfaces which permit local tools to understand formats from other tools, link manager which establishes the communication between the different tools, message board which posts any unresolved errors, and synchronization interface which provides for dynamic integration between the tools.

Witzerman and Nof (1995) present a graphical approach for the development and testing of cell control programs. Also, peripheral device models have been included to create and analyze manufacturing cells.

Cho, Jung, and Kim (1996) present a virtual manufacturing paradigm which includes the following modules: communication media, remote product design system, production planner, generic shop floor control system, remote control and monitoring system. The authors also present several examples of agile manufacturing in Korea, namely in the semiconductor and textile industries, and automotive service.

Other examples of research related to building virtual manufacturing environments include Rockwell's Noumenon architecture (Pierson et al., 1992) and the MKS environment (Glickman et al., 1991). One intent of Noumenon was the design and validation of control software using discrete event simulation. The Noumenon approach to modeling the physical factory is based on an object-oriented graphical model with underlying sequential function charts to represent process behavior. The physical process model in Noumenon depicts the material flow and/or transformation of the process, and the behavior of the sensors and actuators. Noumenon is implemented in Smalltalk-80.

Furthermore, one of the key components of virtual manufacturing is the virtual plant which is a software representation of the physical system. Several authors have discussed the benefits of developing an object-oriented map of an actual processing plant (Stephanopoulos, 1990, Roberts et al., 1991, Rosenberg et al., 1992, Barton and Pantelides, 1993). Although the Instrument Society of America SP88 standard (1993) for batch control systems, models (equipment, recipe, and scheduling), and terminology may provide a foundation for model development, there is no approach for integrating these models. The major benefit of integrating models into a virtual plant is that the physical system components ( i.e., equipment, materials, etc.), conceptual system components (i.e., process plans, equipment schedules, etc.), and

associated characteristics can be easily and naturally represented through the creation of virtual plant entities emulating their structure and function. Thus, the physical and conceptual system entities can be added to and removed from the virtual plant as necessary with minimal impact on the other data in the system.

### 3.2 Multipurpose Plant Scheduling

Numerous authors have developed solution procedures for the scheduling of multipurpose batch plants (Patsidou and Kantor, 1991, Kondili et al., 1993, Dessouky et al., 1996). The disadvantage of all the above approaches to batch chemical scheduling is that they are developed for a static environment. In a static environment orders arrive simultaneously, in contrast to the real-life dynamic environment where orders arrive in sequence, mostly a random one. Applying the static assumption in a dynamic environment, schedules are updated periodically, rather than continuously, with arrival of new orders and the occurrence of changes in the system state occurring during the period. By lumping events that occur over a period of time into one instant the schedule loses some of its realism and effectiveness.

There is a significant amount of work in on-line dynamic scheduling of discrete-parts manufacturing. Harmonosky and Robohn (1991) and Suresh and Chaudhuri (1993) present an excellent review of this work. The authors state that artificial intelligence, math programming, simulation and heuristics are the primary techniques used for real-time dynamic scheduling. However, the literature on discrete-parts manufacturing does not take into consideration the unique characteristics of batch chemical processes. There has been limited work on dynamic scheduling in batch chemical manufacturing. Ishii and Muraki<sup>(Ishii and Muraki, 1996)</sup> present a heuristic approach for multiproduct batch plants. In this environment, all products follow the same sequence of operation. Goodall and Roy (1996) present a methodology to dynamically reconfigure a batch plant when changes in the schedule occur.

### 3.3 Logic Control

A real-time control specification is a formal characterization of the requirements and desired behavior of the system to be controlled. A robust specification is important because it provides a standard mechanism for communication of the intended product. Verification is the

demonstration that the designed logic will perform as intended. If performed after or during implementation, verification has been shown to consume almost 50% of the entire development cost (Halang and. Sacha, 1992).

There are several specification methodologies that take an aggregate view of the system, using device interactions, typically in terms of system or device states and events, that are useful for logic verification and validation. These methodologies are sufficiently robust for actual control. Such methodologies include petri nets and statecharts. Petri nets have been widely used to model the real-time logic of manufacturing systems (Sacha, 1993). Statecharts, which were developed to offset the drawbacks of the finite state machines, have also been demonstrated to be a robust logic specification tool ( Harel, 1990). Both techniques have been used for logic control.

After a specification has been developed and verified, the control logic must be implemented. Most specifications are translated to a vendor specific language. Some authors suggest that the specification should be used for both design and control without major modification<sup>(Halang and. Sacha, 1992)</sup>. While these methodologies can be mapped to an implementation, it is not common practice.

The international standard for PLC programming ( IEC, 1992) includes Sequential Function Charts (SFC) and Function Blocks (FB) that can be used as function-based design tools. These standards are based loosely on petri net theory.

#### 4. Overview of VPMOD

We begin our discussion with an overview of VPMOD. Later sections discuss in detail the models that comprise VPMOD.

VPMOD consists of the following integrated distributed models: plant, product, order, logic controller, and simulation. These models communicate through the model supervisor. Figure 2 shows the information flow and linkage between the various models. The purpose of these models is to represent system states, system events, and predicted future system behavior more accurately and promptly for real-time factory control and scheduling. The models are implemented in an object-oriented environment.

( \*\*\*\*\* Insert Figure 2 here \*\*\*\*\* )

The plant model is a software representation of the physical system. It consists of the process equipment (reactors, columns, pipes, sensors, etc.), the states of the process equipment, and the current state of any product in the process equipment. The plant model states are updated through connection to the simulation model, which acts as the actual physical system. The product model consists of the process recipe(s) for each product to be produced. The process recipe defines the sequence and parameters of the operations necessary to produce the product. The order model consists of product type, quantity, due date, and priority.

The simulation model represents the operation of the plant and is used to generate many of the system events. User initiated events are also captured by the simulation model. These system events include sensor inputs, equipment breakdowns, and input from other models such as predicted process times. Other system events can come from dynamic user interaction with the model.

The model supervisor is a constraint-based scheduling model that can aggregate inputs from many types of models (i.e., plant, product, and simulation). The model supervisor defines and manages the rules to support the overall operational objectives of minimizing time and/or cost and rules for individual batch priorities that can supersede the overall operational objectives. Strategies for batch sizing, batch splitting, batch merging, clean-out processes, maintainability, and variation in batch processes is incorporated in the model supervisor.

One of the responsibilities of the model supervisor is to dynamically modify the formal logic controller as current events (breakdowns, new orders, process terminations) and anticipated future events (scheduled maintenance and predicted process terminations) occur. This approach provides a feedback and a feed-forward control strategy. The feed-forward strategy is important because there are time horizon issues in the link between the formal logic controller and the model supervisor. The model supervisor is also the user interface for interactive schedule modification.

The formal logic controller model represents the sequence control of the headers and processes in the plant. It also represents a generic analog of a vendor-specific logic control system. The control model is a control interpreted petri net, which specifies state transitions (control actions) based on system states and events. The logic controller model is sufficiently generalized to be translatable in the diverse syntax of commercial control systems.

## 5. Plant Model

The plant model is a grouping of process equipment entities (tanks, reactors, separators, mixers, heat exchangers, etc.) that are physically or logically linked. The most common links between equipment entities are physical links such as headers comprised of pumps, valves, and pipes. It is useful to view the plant model as a directed graph, or network, where pipes are arcs between process equipment or nodes. With this distributed configuration, any piece of equipment has access to the other equipment instances through the pipes.

Each node maintains information pertaining to its physical or static condition. This includes vessel name, capacity, material composition, and connected equipment. Each node also maintains information pertaining to its dynamic condition. This includes chemical content, associated order, pressure, temperature, volume, and its current process function. The schedule is also distributed in the network. There are numerous classes and variables that are used to implement the plant model. We now discuss the most salient plant elements.

The plant model,  $P$ , can be depicted as the three tuple:

$$P = \{U, J, O\} \text{ where}$$

$U$  denotes the set of processing units,

$J$  specifies the set of junctions, and

$O$  represents the set of orders.

In this section sets  $U$  and  $J$  are described. The order element  $O$  is discussed in the next section. An instance of set  $U$  is an associated set of equipment usually related to one process, such as a mixer that has one or more pumps, valves, sensors and actuators. This is represented as:

$$U_i = \{E_i, C_i, S_i\}$$

$E_i$  is the set of equipment in unit  $i$ ,

$C_i$  is the set of input and output pipes to unit  $i$ ,

$S_i$  is the schedule for unit  $i$ , and

$\Psi$  is the current function of the unit.

Each equipment set,  $E_i$ , may contain several machines. Thus,  $E_i = \{e_1, e_2, \dots, e_n\}$  and

$$\dots \quad e_\alpha = \{\beta, \dots\}$$

$e_\alpha$  is an individual equipment,

$\alpha$  is the index of the equipment,

$\beta$  is the equipment name,

reactor),

$\gamma$  is the equipment type,  
 $\delta$  is the set of equipment characteristics (eg. glass or steel  
 $\varepsilon$  is the equipment states (volume, temperature and pressure) and  
 $\theta$  is the current order and batch being processed.

The set of input and output pipes to an unit is  $C_i = \{I_i, K_i\}$  where  $I_i$  is the set of input pipes and  $K_i$  is the set of output pipes. Each unit contains its own schedule, which is represented as a four tuple:

$$S_i = \{\kappa, \tau, \upsilon, \omega\} \text{ where}$$

$\kappa$  is the list of scheduled batches for the unit,

$\tau$  is the list of batch sizes,

$\upsilon$  is the list of scheduled start times for each batch, and

$\omega$  is the list of scheduled stop times for each batch.

The unit function  $\Psi$  is one of the following values: transferring, processing or waiting.

The set of junctions  $J$  is a special case of the equipment type  $e_\alpha$ . Junctions are also called headers. This equipment type is a switch for many input and output pipes where only one concurrent interconnect is possible.

Many functions, or methods, have been created to assist in the development of plant models. Most of these functions aid in rendering and maintaining model links. In the object oriented implementation the elements in sets (U, J, O, E, C, S, K, I) are implemented as classes. The values represented by the greek characters are implemented as instance variables of an instantiated class. This convention is used throughout this discussion. The plant model development interface is shown in Figure 3 for the process plant in Figure 1.

( \*\*\*\*\* Insert Figure 3 here \*\*\*\*\* )

In Figure 3 the pallet of process units, junctions and pipes used to create the plant model elements are shown. Elements are selected and placed in the plant modeling area of the screen. Pipes are connected to the units and junctions by selecting the pipe icon and then selecting both

of the connecting units or junctions. The user can move a unit by dragging it to a new location. Pipe connections are maintained during drag operations. Instance variables are entered by selecting the appropriate unit or equipment from the modeling area where an input screen will appear for the equipment variables.

The physical plant models are maintained as an object. This permits the creation and maintenance of many concurrent plant models. The main objective of the plant model is to simulate the function of the actual plant. The plant model states are updated through connection to the simulation model or ideally through the actual physical system. We next discuss the product and order models.

## 6. Order and Product Models

An order model contains attributes that pertain to a customer order. Let  $O = \{O_1, O_2, \dots, O_m\}$  be the set of orders. Each order,  $O_k$ , in set  $O$  is represented as a four tuple:

$$O_k = \{\eta, \lambda, \mu, \pi\} \text{ where}$$

$\eta$  is the product name,

$\lambda$  is the order due date,

$\mu$  is the order quantity and

$\pi$  is the order priority.

A product model represents the high-level recipes of the product (ie., ammonia, margarine) that is to be manufactured. The product model of type  $l$ ,  $R_l$ , is represented as the tuple:

$$R_l = \{H_l, Q_l\} \text{ where}$$

$H_l$  is the set of input chemicals, and

$Q_l$  is the ordered set of process steps.

The high-level recipe defines the required input chemicals as the tuple:

$$H_l = \{\rho_l, \sigma_l\} \text{ where}$$

$\rho_l$  is a list of input chemical types and

$\sigma_l$  is a list of input chemical quantities.

The high-level recipe defines the required process steps as an ordered sequence:

$$Q_i = \{q_1, q_2, \dots, q_n\} \text{ where } q \in U^*$$

$U^*$  is a type of unit that exist in the plant model.

It is not necessary to populate all of the instance variables for  $U^*$ . Instance variables that are not populated imply that any instance variable is acceptable for the process step. Most frequently the instance variable  $\delta \in e_\alpha$  is defined to distinguish between processes and units. For example, some of the reactors are glass lined and some are steel lined for the example plant. If the reaction process step required a special type of reactor such as a glass lined reactor,  $\delta$  would be entered.

There may be more than one recipe for a particular product. Process recipes are constructed in a multi-level fashion, therefore, a particular process recipe contains an ordered set of sub-processes that can be used to produce a particular chemical batch. Figure 4 depicts the graphical depiction for one developed recipe for the example plant.

( \*\*\*\*\* Insert Figure 4 here \*\*\*\*\* )

In this figure there are two input chemicals where the recipe calls for a 60% concentration of chemical A mixed with a 40% concentration of chemical B. The mixing time for each chemical is 5 minutes. The reaction is to take place in a glass reactor. An approximate reaction time of 30 minutes is specified for the reaction process. The actual reaction process ending time is stochastic event. The instance variables that are populated in the high-level recipe are shown below the process step icon.

## 7. Logic Control Model

Schedules are insufficient for actual transport control of a plant since schedules represent deterministic process times and real system performance is stochastic. Thus as process variation occurs a schedule becomes unsynchronized with the actual process. Transport of the chemicals in the process plant is performed by a logic controller that is typically part of the entire process control system, which is also responsible for unit control where setpoints are maintained. Although control logic is generally implemented in a vendor specific language, we use a control

interpreted petri net (David and Alla, 1992) as a generic discrete event logic model that can be mapped to vendor specific languages. The current IEC 1131 standard for programmable logic control (IEC, 1992) specifies several options for discrete event control logic development, including a Sequential Function Chart (SFC), which is a variant of a petri net. The general functioning principle of a petri net is that places or nodes contain tokens that pass through connected transitions to other nodes when the transition is enabled.

The logic control model,  $L$ , can be represented as the eight tuple:

$$L = \{A, T, Pre, Post, \Gamma, \Lambda, \Sigma, \Omega\} \text{ where}$$

$A$  is a set of places (nodes),  $A = \{A_1, A_2, \dots, A_r\}$ ,

$T$  is a set of transitions,  $T = \{T_1, T_2, \dots, T_p\}$ ,

Pre:  $A \times T \in \{0, 1\}$  where 1 means  $A$  is an input to  $T$

Post:  $A \times T \in \{0, 1\}$  where 1 means  $T$  is an input to  $A$

$\Gamma_i$  is the timing associated with a place  $A_i$

$\Lambda_i$  is the operation to be preformed at  $A_i$

$\Sigma_j$  is the event at  $T_j$  that enables the transition and

$\Omega_j$  is the condition at  $T_j$  that enables the transition.

In this specification the nodes,  $A$ , also called places are the states in the system, including processing and transferring activities. A token moves through a transition if the conditions,  $\Omega$ , are satisfied and when the event,  $\Sigma_i$ , transpires. There may be no conditions or no event specified, which relieves this requirement. Upon arriving at a new place the operations,  $\Lambda$ , are carried out. The token is then available for further processing after waiting for the timing delay,  $\Gamma$ .

Figure 5 depicts a segment of a logic model for the example plant for batch flow control between tank A2 and mixer M3 from Figure 1. Note on Figure 1 that there are variables associated with the valves that are controlled in Figure 5. These are represented as v1, v2 and so on. The token in place P1 will move when event  $E^3$  occurs (request for batch loading). The token (represented by the darkened circle in Figure 5) will move to P3 if valves v3, v4, v5, v8, and v10 are closed. This is to prevent flow of the chemical into other pipes. If any of these

valves are open then the token moves to place P2 where an error notification is given. Event E<sup>9</sup> is an operator input that clears the system for retry. If the token moves to place P3 then a control action is taken to open valves is sent to valves v2, v6, v7, and v11. The token is then delayed for two seconds ( $d2=2s$ ). If the valves have successfully opened the token will move to P5 where mixer 3 (M3) will be turned on. If the valves have not responded the token will flow to P4 where an error notification is generated.

(\*\*\*\*\* Figure 5 here \*\*\*\*\*)

The logic control model is a formal representation of the sequential logic that is used to control the transfer of products through a batch plant. The model is integrated with the simulation and plant models. The simulation calendar also includes control events. Thus, when an event transpires, the logic control model is executed to determine future events. These events then are placed on the simulation calendar. The structure and parameters of the logic control model are dynamically changed by the model supervisor.

## 8. Simulation Model

Inputs to the simulation model come from the formal logic controller model and simulation model parameters that represent operation times, state equations, breakdowns and order arrival. In a high-level perspective of the simulation model, physical objects such as products and equipment from the plant model are connected to the simulator. The simulation experiment objects represent the parameters for one experiment for the products, the equipment, and the behavior of the virtual plant. These can be changed for each simulation. Simulation controller objects coordinate the execution of time, events, and input/output to the simulation from an external source. At initialization, the data from the experiment objects are transferred to their corresponding physical objects. During the simulation execution, the simulation objects use the data from the external inputs to drive the simulation model in the physical objects. Data is transferred between the physical and simulation objects.

The simulation experiment object holds the information for the entire simulation experiment. Let  $n$  be the number of individual pieces of equipment defined in the experiment,

and let  $m$  be the number of products defined in the experiment. The simulation experiment object is represented as the tuple:

$$X_s = \{Y_s, Z_s\} \text{ where}$$

$Y_s$  is the set of individual equipment defined in the experiment

$$(Y_s = \{e_1, e_2, \dots, e_n\}), \text{ and}$$

$Z_s$  is the set of products defined in the experiment

$$(Z_s = \{R_1, R_2, \dots, R_m\}).$$

Each individual piece of equipment,  $e_\alpha$ , and product,  $R_i$ , has a set of state variables,  $W$ , where  $W = \{w_1, w_2, w_3, \dots, w_p\}$ .

Each state variable  $w_\Delta$ , where  $\Delta$  is the index for the state variable, changes its value over time. The state variable has the following attributes  $\{\gamma, \xi, \phi, \varpi, \varphi\}$  where

$\gamma$  is the state variable name,

$\xi$  is the process equation which models the behavior of the variable,

$\phi$  is the integration algorithm used to integrate the process equation,

$\varpi$  is the local step size used to update the state variable, and

$\varphi$  is the set of parameters that define the state-event (eg. tolerance, direction).

The simulation controller object controls the execution of the simulation and in most cases the communication between the other models. An instance of the simulation controller is created for a simulation run. This instance is a global variable. Thus, every object in the simulation has access to the simulation controller object. The simulation controller object is represented as the tuple:

$$F = \{K, M, N\} \text{ where}$$

$K$  is the set of parameters that define a simulation calendar (eg. current time, g

$M$  is the set of parameters which permit communication with .... external sys

$N$  is the random variable generator (eg., normal, uniform, exponential).

The simulator uses a next event approach where events are placed on a centralized calendar. Difference and differential equations are also updated at discrete intervals. When the

event is triggered those state equations that should be updated are. If these equations trigger other events they are also placed on the calendar.

A major benefit of this simulation environment is the distributed nature of the objects. In traditional simulation languages the equations for the state variables are all located in one centralized subroutine. In those cases, the user is restricted to a single step size and integration algorithm for all the state variables. Code reuse is limited in this case because any change to the model requires a new subroutine to be written. A complete discussion of this simulation modelling approach can be found in Dessouky et al. (1995) .

## 9. Model Supervisor

The model supervisor is the link between all of the models. Its primary function is to monitor system events and to transform information from the system events and each of the models into new or updated models. There are several system events that can induce model changes. These include user requests, change or arrival of a new order, completion of operation processing, completion of batch transfer, batch failure, equipment failure, and process attribute updates. In this prototype system the simulation model and the user generate all of these events. Actual plant sensors and actuators would ultimately replace many of the simulation model functions. Figure 6 shows the user interface to the supervisor. The user interface requests are shown in the figure. These include Order for creating orders, Initial for generating the initial schedule, Proposed Schedule for generating reactive schedules and replacing the current schedule, Control Setup for generating and assigning control logic, Simulation for simulating the plant, and Output for generating results.

\*\*\*\* (Insert Figure 6 here)

### 9.1 Generate Process Routings

The primary link between the plant model and the product model is a supervisor function that provides the ability to match products to processes. This is represented as the mapping of the recipe onto the plant:

$M: (R_i \times P) \rightarrow V_i$  where  $M$  is a mapping function and  $V_i$  is the set of possible process routings for recipe  $R_i$ . The purpose of this mapping function (also called the matcher) is to

identify all possible routings and equipment that can be used to make the product. A specification of the units that can satisfy each operation in a product's process recipe is referred to as a process route. The matcher views the plant model as a network where the pipes are the arcs in the network and the other process equipment are the nodes. The network can be traversed through the input and output connectors that are the attributes of an equipment entity. The matcher equates the equipment type from the process recipe to specific units from the plant model. When a match for one unit type is found it is identified and stored as a process chain. The mapping function then searches for the next sequential process step in the recipe by looking at all connecting nodes to the chain in the network. If a match is found then the equipment is added to the process chain. When the entire network has been traversed, incomplete chains are discarded. It is possible to have the matcher return many, one or no matching chains. These chains are used for scheduling and dispatching functions.

## 9.2 Generation of a Schedule

The schedule is important because it provides the basis for control of the plant. Schedules are generated at the beginning of plant operation and then again when system events occur that make a new schedule more favorable. Such events include change or arrival of a new order, batch failure, equipment failure and a significant departure between the predicted schedule and the actual plant operation.

The inherent flexibility of multipurpose batch chemical plants (i.e., alternative equipment routings) offers the possibility of dynamically changing the schedule and correspondingly, the process routings. A new schedule must consider in process batches and the consequences of re-routing these batches. Some chemical products have strict waiting and post reaction time constraints. Re-routing might also necessitate equipment cleanout depending on the prior product in the equipment. The model supervisor is responsible for selecting the appropriate manufacturing alternative based on the current state of the equipment sets in the factory and the location of the batches. Strategies for batch splitting, batch merging and clean-out have been incorporated into the supervisor.

The generation of a schedule is a mapping:

$$M: (V(O_y) \times (\hat{U} \vee \hat{J})) \rightarrow S_y, \quad \forall y \text{ where}$$

$V(O_y)$  is the process routings for chemical order  $y$ ,

$\hat{U}$  is the set of available units,  
 $\hat{J}$  is the set of available junctions and  
 $S_y$  is the schedule of batches for order  $y$ .

The scheduling approach taken in VPMOD is applicable to multipurpose batch chemical plants. In these plants, all product types may not necessarily follow the same sequence of operations. VPMOD produces a deterministic schedule from which control logic is developed. The control logic follows the sequence of batch assignments to control product release and flow. However, the logic is designed for the stochastic nature of the actual process operation. The actual transfer times and process times will not exactly match the deterministic schedule. When the departure between the deterministic schedule and the actual operation reaches a predefined limit, or at the user discretion, a new schedule is created. By comparing schedule metrics of lateness, equipment utilization, and/or production cost the new schedule can be implemented by generating new control logic. This is also the case for equipment breakdown.

In VPMOD, the schedule is developed to take into consideration the special characteristics of this environment such as continuous entity, batch sizing, and waiting time limit. Additionally scheduling rules determine the batch sizes and the timing of the release of new batches. This approach minimizes any possible waiting time violations due to potential plant congestion, and hence reduces the risk of spoilage.

Rules are also developed to determine which batch is processed next when an equipment set becomes idle. VPMOD uses the least-slack dispatching rule to determine which customer order is processed next, where the slack is defined to be the difference between the expected completion time of the order and the due date of the order, in order to minimize the total tardiness. The least slack rule is used because this rule gives highest priority to an order which has the shortest time remaining to the due date (Morton and Pentico, 1993). Other heuristics were tested such as first come first serve, earliest due date, and shortest processing. However, these rules were poor performers in minimizing total tardiness. These results are not surprising because none of these rules take into account the time required to complete the finished product or the due date of each order during the scheduling process.

After using the least slack rule to determine which customer order is scheduled next, VPMOD schedules the batches that will satisfy the demand for that order on the machines that

will complete processing it the earliest based on the current factory status. The batch size is then set to the smallest capacity of any machine in its sequence of operations. A complete discussion of these scheduling priority rules can be found in Dessouky et al. (1996). VPMOD produces a Gantt chart of the process function across the equipment. This is shown in Figure 7. VPMOD optionally produces a scheduling view by batches, which is particularly useful for batch tracing.

( \*\*\*\*\* Figure 7 \*\*\*\*\* )

Another scheduling feature permits the user to force changes in the generated schedule by dragging a batch icon from one location on the Gantt chart to another. Functions have been developed that verify the appropriateness of the move and recalculate the processing times based on unit capacity. The remainder of the schedule is modified using a deviation propagation tree that identifies conflicts (ie., overlapping batch processes and transfers). The tree identifies the first conflict and reschedules the overlapping batch, which in turn can produce additional conflicts. This procedure continues until all conflicts are resolved. A complete discussion of this approach can be found in Lee (1996).

### 9.3 Generation of Control Logic

Formal logic control models are generated based on the schedule found in the plant model. The control logic generation includes parameter as well as structural changes in the model. Our approach to the logic controller model generation identifies primitives from the control net that map to batch control actions. A rule-based system is used to construct higher-level sets of the primitives that compose the sub-control networks, such as transfer from one location to the next that might involve several headers, pumps, and pipes.

The control logic is derived from the sets of scheduled batches,  $S$ , determined by the scheduling function. Inherent in the scheduled batches is the sequence of the batches and the equipment that the batch will use. While the schedule is deterministic, the actual control must be able to accommodate stochastic process and transfer times. The control logic maintains the predetermined sequence and uses events instead of time for flow control. The generation of control logic is a mapping:

$M:(S \times T) \rightarrow L$  where  $T$  is the real stochastic operating space.

Recall that  $L$  is the logic model which consists of nodes  $A$ , transitions  $T$ , operations defining relations between the nodes and transitions  $Pre$  and  $Post$ , timing  $\Gamma$ , operations  $\Lambda$ , events  $\Sigma$ , and conditions  $\Omega$ .

Timing,  $\Gamma$ , is used when it is necessary to delay an operation. Operations,  $\Lambda$ , represent the connection between the control model,  $L$ , and the actuators on the physical equipment. These are all considered to be discrete. Some of the operations include start process, end process, open valve, close valve, turn on pump, turn off pump. Events,  $\Sigma$ , represent the input from the equipment to the control model. Some of the events include operation complete, operation started, valve closed, valve opened, pump on, pump off. Conditions,  $\Omega$ , represent the interlocks between the equipment. Conditions are essentially states or variables that govern operations. An example of a condition for turn on pump is: If Valve is Open.

The control logic operates in real time if connected to equipment or simulated time if connected to the simulation. During operation an operator would choose to reschedule when a proposed schedule produced better operational statistics (produced by the Stat button). When this is done the operator must also generate new control logic and assign the logic before the new logic functions. Safeguards have been placed in the assign function that prevent assignment during a transfer. There is no output view of the control logic since it is text based. Graphics are planned for future work.

## 10. Conclusion

We have presented a VIRTUAL PLANT MODELER, VPMOD, for design and control of batch chemical product flow. VPMOD consists of several integrated formal models. These models include a plant design model, a chemical recipe model, a schedule, a logic control model, and a simulation model. These models provide a framework for agile batch chemical manufacturing that have the ability to automatically reroute and control chemical product flow in a flexible plant subject to unexpected events and varying demand patterns. The models have been integrated in an object-oriented paradigm and tested on an industry provided flexible batch plant.

## References

- Baudel, B. Cantgrit, E., and Toulette, J.M., (1988) "Smalltalk and Simulation of Batch Dynamic Simulator for Training (SIDEN)", *Computers in Industry*, 8, pp325-337.
- Barton, P.I. and Pantelides, C. C., (1993) "gProms - A Combined Discrete/Continuous Modelling Environment for Chemical Processing Systems,"*International Conference on Simulation in Engineering Education*, La Jolla, CA , 25:3, pp25-34.
- Cho, H., Jung, M. and Kim, M., (1996) "Enabling Technologies of Agile Manufacturing and Its Related Activities", *International Journal of Production Research*, 34:2, pp329-348.
- David, R and. Alla, H (1992) *Petri Nets and Grafset*, Prentice Hall.
- Dessouky, Y.M., Roberts, C.A., Dessouky, M.M. and Wilson, G. (1996) "A Heuristic-based Scheduler for a Flexible Manufacturing System", *International Journal of Production Research*, 34:2, pp329-348.
- Dessouky, Y.M. Roberts, C.A. and Beaumariage, T.G. (1995) "Object-Oriented Simulation Architecture for Flexible Manufacturing Systems", *International Journal of Production Research*, 33:12, pp1251-1268.
- Glickman, J., Hitson, B.L., Pan, J.C. and Tenenbaum, J.M., (1991) "MKS: A Conceptually Centralized Knowledge Service for Distributed CIM Environments," *Journal of Intelligent Manufacturing*, 2, pp27-42.
- Goodall, W.R. and. Roy, E (1996) "Short Term Scheduling and Control in the Batch Process Industry Using Hybrid Knowledge Based Simulation," *International Journal of Production Research* , 34:1, pp33-50.
- Halang, W and. Sacha, K, (1992) "Achieving high integrity of process control software by graphical design and formal verification," *Software Engineering Journal*, 7 (1), pp53-64.
- Harel, D (1990) "Statemate: a working environment for the development of complex reactive systems," *IEEE Transactions on Software Engineering*, 16 (4), pp403-414.
- Harmonosky, C.M. And Robohn, S.F. (1991) "Real-time Scheduling in Computer Integrated Manufacturing: A Review of Recent Research," *International Journal of Computer Integrated Manufacturing*, 4, pp331-340.
- Hofman, J.K. (1993) "Agile Manufacturing In The Process Industries", *Journal A*, 34:3, pp47-50.
- IEC 1131, (1992) *International Standard on Programmable Controllers Part 3: Programming Languages*, International Electrotechnical Commission.
- Instrument Society of America, (1993) "Batch Control Systems Models and Terminology," *ISA SP88*.
- Ishii, N. and. Muraki, M, (1996) "An Extended Rule Approach in an On-Line Scheduling Framework for Batch Process Management," *International Journal of Production Research* , 34:2, pp329-348.
- Kondili, E., Pantelides, C.C., and Sargent, R.W. (1993) "A General Algorithm for Short-term Scheduling of Batch Operations - MILP Formulation," *Computers & Chemical Engineering*, 17, pp211-244.
- Lee, Y.J. (1996) "Interactive Reactive Schedule Modification of Batch Non-Discrete Entity Manufacturing Systems," *An Unpublished Ph.D. Dissertation*, Arizona State University, Tempe, AZ.
- McCarthy, J.J. (1990) "The CIM Reference Model as a Tool for Plant Information and Control Systems Development," Internal Honeywell Paper, Honeywell, Inc. I.A.C.D., Phoenix, AZ.
- McGehee, J., Hebley, J. and Mahaffey, J. (1994) "The MMST Computer-Integrated Manufacturing System Framework," *IEEE Transactions on Semiconductor Manufacturing*, 7:2, pp107-116.
- Morton, T.E. and Pentico, T.E., (1993) *Heuristic Scheduling Systems*, John Wiley & Sons, NY.
- Musier, R.F. and Evans, L.B, (1990) "Batch Process Management", *Chemical Engineering Progress*, June, pp66-77.

- Parks, C. M. , Koonce, D.A., Rabelo, L.C. Judd, R.P., and Sauter, J.A., (1994) "Model Based Manufacturing Integration: A Paradigm for Virtual Manufacturing Systems Engineering," *Computers and Industrial Engineering*, 27, pp357-360
- Patsidou, E.P. and Kantor, J.C., (1991) "Scheduling of a Multipurpose Batch Plant Using a Graphically Derived Mixed-integer Program Model," *Industrial and Chemical Engineering Research*, 30, pp1548-1561.
- Pierson, B.L., Morely, D.J., Agre, J.R. and Clare, L.P., (1992) "Noumenon: A Factory Design Environment", Proceedings of the 1992 Object Oriented Simulation Conference, La Jolla, CA., SCS Publications, pp1-15.
- Rippin, D.W. (1983) "Design and Operation of Multiproduct and Multipurpose Batch Chemical Plants - An Analysis of Problem Structure," *Computers & Chemical Engineering*, 7, pp.463-481.
- Roberts, C.A., Beaumariage, T.G., Dessouky, Y.M. and M.K. Ogle, (1991) "Object-oriented Simulation Tools Necessary for a Flexible Batch Process Management Architecture", *Proceedings of the 1991 Winter Simulation Conference*, Phoenix, AZ, pp323-330.
- Rosenberg, R.C., Whitesell, J. and Reid, J., (1992) "Extendable Simulation Software for Dynamic Systems", *Simulation*, 58, pp175-183.
- Sacha, K (1993) "Real-time Specification Using Petri Nets," *Microprocessing and microprogramming*, 38, pp607-614.
- Stephanopoulos, G. (1990) "Artificial Intelligence in Process Engineering - Current State and Future Trends," *Computers & Chemical Engineering*, 14, pp1259-1270.
- Suresh, V. and Chaudhuri, D. (1993) "Dynamic Scheduling - A Survey of Research," *International Journal of Production Economics*, 32, pp53-63.
- Suydam, W., (1987) "CASE Makes Strides Toward Automated Software Development", *Computer Design*, January, pp49-51.
- Witzerman, J.P. and Nof, S.Y., (1995) "Integration of Simulation and Emulation with Graphical Design for the Development of Cell Control Programs," *International Journal of Production Research*, 33:11, pp3193-3206.





Figure 1. Configuration of a Test Batch Chemical Plant

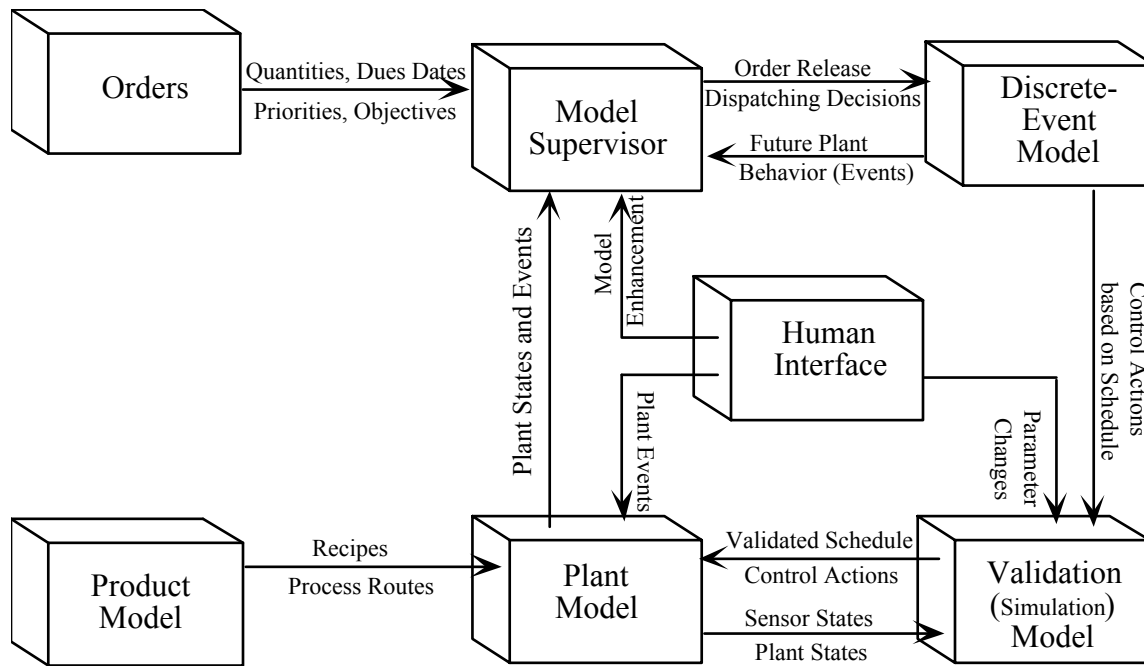
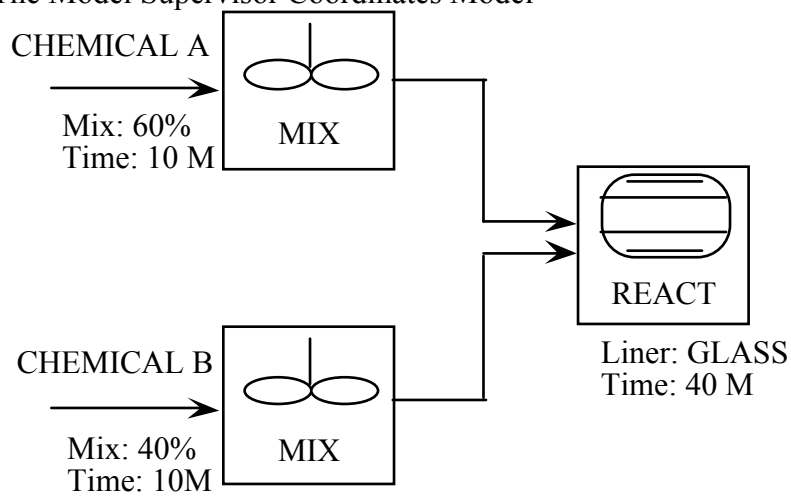


Figure 2. The Model Supervisor Coordinates Model



Interaction Figure 4. A Graphical Recipe Model Specifies the Production Plan

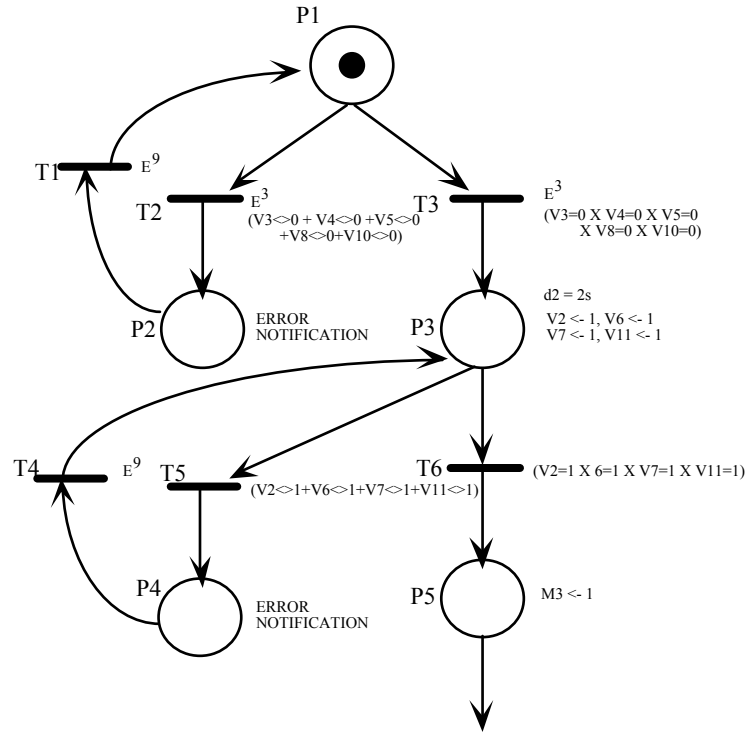


Figure 5 Logic Control Model for Mixer Filling Operation from the Virtual Plant

