Proceedings of the 2011 Winter Simulation Conference
S. Jain, R.R. Creasey, J. Himmelspach, K.P. White, and M. Fu, eds.

GENERIC FRAMEWORK FOR SIMULATING NETWORKS USING RULE-BASED QUEUE AND RESOURCE-TASK NETWORK

Naoko Akiya

The Dow Chemical Company
2301 N. Brazosport Blvd.
Freeport, TX 77451, USA

Scott Bury
John M. Wassick

The Dow Chemical Company
Research Campus 1776 Bldg.
Midland, MI 48667, USA

ABSTRACT

A generic model is a model that is built for a class of system and can be implemented for a specific system through changes in input data alone, without any structural changes to the model. In this paper, we propose a framework for building such generic model for non-steady state process networks, which are characterized by flow of materials between interconnected nodes. The framework comprises two elements: (1) a generic representation of process network structure using a rule-based queue and (2) a generic representation of non-steady state operations of the network using recipe tables inspired by the Resource-Task Network representation. In this paper, we describe conceptually the data structure and simulation logic that can be used to implement this framework in any simulation software. Examples are provided in the context of batch plant operation.

1 INTRODUCTION

Discrete event simulation is a useful tool for analyzing a system, the operation of which can be described as a chronological sequence of events, for the purpose of design validation or improvement. We propose a method of building a generic simulation model that applies to any system that can be regarded as a non-steady state process network and enables rapid execution of design and improvement projects. Examples of process network include batch or discrete manufacturing plant (a network of production units connected by material handling equipment or operator movements), an integrated chemical complex (a network of plants connected by pipelines or tank trucks), or distribution network (a network of warehouses and distribution centers connected by transportation routes). Although each network is unique, all process networks function by transferring materials from one set of nodes to another set of nodes, with time delays at the nodes and along the arcs. In this paper, we use the term "generic" to refer to a simulation model that can be implemented for a specific system largely through input data alone, with minimal or no changes to the model structure. In other words, it is a single model applied to multiple systems, rather than a single model applied to multiple scenarios of a single system.

This definition of "generic" is somewhat different from "generic" as flexibility for re-use as described in Brown's paper (Brown 2010), which also provides a survey of the previous literature and the arguments for and against model re-use, especially in the defense area. The main message of that paper is that model developers gain dividends for themselves and clients with proper planning and looking at the simulation from a data-driven point of view. In this work, we have gone beyond philosophy and describe an actual modeling construct for generic modeling. Our definition of "generic" also differs from a framework of model generation using a general-purpose modeling language, e.g., SysML (McGinnis and Ustan 2009, Schönherr and Rose 2009).

From our point of view, there are two challenges to developing a generic simulation of a non-steady state process network: the generic representation of the network itself and the generic representation of the time-dependent operation of the network. In the proposed approach, these challenges are addressed through the use of rule-based queues and Resource-Task Network (RTN) representations (Pantelides 1994). For the purpose of illustrating this method, we present it in detail for simulating a batch plant. Applications to other types of systems are described conceptually.

2 CONCEPTUAL FRAMEWORK OF GENERIC SIMULATION

A network is characterized by the following basic elements:

- Individual nodes comprising the network
- Arcs connecting the nodes
- Flow of materials at each arc (quantity and timing)

Our view of "generic" simulation is that the same simulation logic can be used to model any number of nodes with any arbitrary connectivity, whether the flow is continuous or intermittent and event-driven.

2.1 Representation of Network Structure for a Batch Manufacturing System

The first element of our approach is the use of rules to represent the network connectivity. These rules specify the criteria that need to be satisfied for the flow to occur from one node to another. Figure 1 illustrates the conceptual difference between non-generic and generic approaches using a hypothetical batch plant comprising three process steps. The non-generic approach is characterized by the use of simulation logic that is structured to emulate the network structure. The items simulated are individual batches of products. Queues are used between process steps for work-in-process (WIP) batches. If the batch does not remain intact throughout the network, unbatching and/or batching of items become necessary. In the generic approach, in contrast, the items simulated are the individual process steps, or nodes in the network. The aforementioned rules are used to control the timing of release of each item from a single queue, which sets the flow between nodes in terms of the timing of batch execution at each process step. In other words, the network structure is something that emerges when the rules are implemented. WIP is modeled by inventory balance, which allows for straight forward mass balance even when the product batch does not remain intact throughout the network.

For the sake of illustration, let us consider batch processes A and B (Figure 2). In this example, each "process" corresponds to a node in a network, which represents an entire plant. The existence of buffers affects the start and finish of a batch. If there is no buffer between A and B, then the following rules generally apply:

1. B starts immediately after A is finished.
2. A can finish only if B is available.

A buffer effectively de-couples the two processes that the above rules no longer apply. Instead, different rules involving the buffer apply:

3. A can run as long as there is capacity in C.
4. B can run as long as there is sufficient material in C.

When buffers are involved, we keep track of their inventories throughout simulation to evaluate the criteria regarding buffer level and capacity.

Throughout this paper, the proposed approach is described using a batch plant as an example, but the approach is applicable to other types of networks, too. Specifically, all networks can be conceptualized to have material movements that occur in batches. Although the term "batch" has a specific meaning in manufacturing, it can be regarded as any discrete form in which the material moves within a specific network. Even if the flow is continuous, it can still be approximated as a series of batch transfers, as long as the flow is discretized such that the simulation results have sufficient resolution to satisfy the simulation objective.

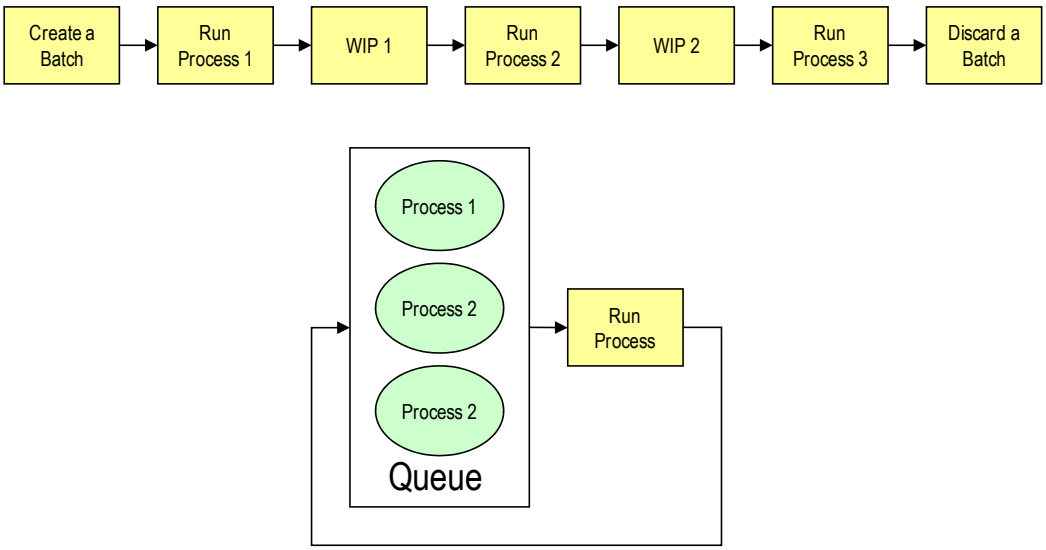

Figure 1: Conceptual difference between non-generic (top) and generic (bottom) models

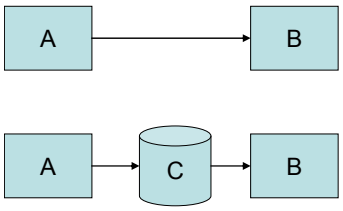

Figure 2: Illustrative examples – two plants without (top) and with (bottom) a buffer

Table 1 summarizes the characteristics of additional network examples. Note that entities that actually move materials (e.g., material handling system, tank trucks) are also considered as nodes in addition to sources and sinks of materials. Such entities should be included in the model if a material transfer step is necessary to make the material from a source available for a sink (as opposed to having a storage tank that is directly accessible by both the source and the sink). Although there are physical differences between different types of networks, the flow between nodes can be controlled by rules similar if not identical to the ones described for the example batch processes in Figure 2. (The reorder point rule is a variation of rule #3 provided for this example.)

In all simulation approaches found in the literature, the process network is built from objects representing all or parts of a unit operation. Each object is generic in a sense that it can be customized by input data. When the model is built, however, the connectivity of those objects is fixed. In other words, once a model is built for plant X, it can be used to model only plant X, not plant Y. This limitation applies even to so called flexible or general-purpose models, which are essentially model generators (see for example, Yuan, Dogan, and Viegelahn 1993; Selvaraj, Hua, and Hess 2005). What is unique about the proposed approach is that, because the process network is specified with input data alone, a single model can be used to model any plant by including the appropriate input data file.

2.2 Representation of Resource Balance Over Time

The second element of the proposed approach is the use of recipe tables to represent the operation of each node as resource balance over time. This capability is critical for modeling nodes with non-steady state operation, such as a unit operation in a batch plant or a workstation in a discrete manufacturing plant. At different points in time during a batch, different materials may be consumed or generated, and different

pieces of equipment may be used. These details are specified in a recipe, which is unique for each product and process. Although both batch and continuous plants use operators, what is unique to a batch plant in terms of operator involvement is that operator tasks are often part of the production sequence. Examples include moving WIP parts from one place to another, loading a machine with the necessary parts and materials, and performing sampling and analysis for the purpose of product quality control. Production cannot continue until these tasks are completed. Timely execution of process steps, which depends on the availability of resources (e.g., material, equipment, labor), impacts throughput.

Table 1: Characteristics of different types of networks

Attributes	**Discrete Manufacturing Plant**	**Integrated Chemical Complex**	**Distribution Network**
Nodes	• Workstations inside plant • Material handling system • Operators with forklifts, carts, etc.	• Plants inside complex • Inter-tank transfer system • Tank trucks	• Distribution centers • Warehouses • Various modes of transportation
Arcs	• Paths followed by parts & products	• Paths followed by bulk chemical (pipeline, truck routes)	• Transportation routes followed by shipments
Example rules for flow	• Similar to batch plant with and/or without buffers	• Similar to batch plant with buffers	• Reorder point
Example of "batch"	• A set of parts to be loaded to a machine • A cart full of WIP products	• Arbitrary discretization of continuous flow per plant • A quantity of chemical transferred between plants (inter-tank transfer or tank truck)	• Individual shipments

The use of recipes to describe batch unit operations is not a new concept. Process simulators such as Aspen Batch Process Developer (AspenTech 2011) or gPROMS (Winkel et al. 1995) are recipe driven. In this case, the recipes are customized for specific unit operations that are available in the tool. Any deviation of a real plant from the templates provided cannot be modeled. Discrete event simulation is a general simulation tool, so recipe structures depend on what the modelers design (see Bernal-Haro et al. 2002 for example). What is unique about our approach is the use of recipe tables derived from Resource-Task Network (RTN) (Pantelides 1994), which allows specifications of the timing of various resource consumption and generation (i.e., not just their quantities), as well as those of operator tasks, within a single task representing a unit operation.

RTN is a mathematical representation of a system originally developed for and that continues to be used for scheduling optimization. RTN describes the process of interest as a network of resources and tasks that interact with each other. In this context, a task is any activity that consumes and/or generates a specific set of resources, including materials (feeds, products, byproducts, utility, etc.), equipment, and manpower. At The Dow Chemical Company, RTN is the basis for a generic scheduling optimization application (Wassick and Ferrio 2011), developed internally, that can be customized for each plant strictly through input data, as we are striving to do presently with discrete event simulation. In a Winter Sim case study we presented the parallels between the discrete event simulation and RTN and how they can complement each other (Bury, Akiya, and Ferrio 2004). Here we have moved RTN into the discrete event simulation as part of the modeling framework.

In the original formulation of RTN, a general mass balance for a resource r is expressed as

$$R_{r,t} = R_{r,t-1} + \sum_{k} \sum_{\theta=0}^{\tau_k} \left(\mu_{kr\theta} N_{k,t-\theta} + \nu_{kr\theta} \xi_{k,t-\theta} \right) + \Pi_{rt} \quad \forall r,t \,, \tag{1}$$

where $R_{r,t}$ is the amount of excess resource r available at time t. The term in parenthesis represents the amount of resource r generated by task k at time t. The parameter θ represents time relative to task start, where $\theta = 0$ marks the beginning of the task k and $\theta = \tau_k$ marks the end of the task. So resources may be consumed or generated at any point during the task (to the extent that it is possible with chosen resolution in time discretization). The parameter $\mu_{kr\theta}$ is the amount of resource r consumed (if $\mu < 0$) or generated (if $\mu > 0$) per occurrence of the task k at time t. The binary variable $N_{k,t}$ has the value of one if the task k starts at time t (only at $\theta = 0$). The parameter $\nu_{kr\theta}$ is the amount of resource r consumed (if $\nu < 0$) or generated (if $\nu > 0$) relative to ξ_{kt}, the extent of task k at time t. The task extent is a characteristic quantity associated with the task, such as batch size of a batch process. The term Π_{rt} represents external resource transfers (i.e., not consumed or generated by tasks).

A recipe table can be derived from RTN, as illustrated by Table 2 for an example task representing a batch unit operation. "Fraction of Batch Completion" is equivalent to θ in RTN. This parameter is used instead of the actual θ to make it easier to change the batch time independently of the recipe. (For a complex recipe, however, it is likely that this parameter needs to be adjusted if a batch time is modified.) "Equipment" is any resource whose quantity changes by μ ("Change in Equipment Quantity"). "Inventory" is any resource whose quantity changes by ν ("Change in Inventory Level as Fraction of Batch Size") times ξ (batch size). Table 2 is just one example of adapting RTN to a recipe table for time-dependent operations. Other forms are possible.

Table 2: Example process recipe table using RTN

Step ID	Fraction of Batch Completion	Equipment	Change in Equipment Quantity	Inventory	Change in Inventory Level as Fraction of Batch Size
1	0	Reactor	-1	Raw Material	-1
2	0.5				
3	1	Reactor	1	Product	1

Besides the ability to model time-dependent operations, another advantage of RTN is that it is general. Resources included in RTN need not always be physical equipment or materials in inventory but can also be an abstract entity representing a condition or criterion that needs to be satisfied. Tasks need not always be batch processes or material transfer but any transaction that consumes and/or generates these resources. This generality also applies to the RTN-inspired recipes in discrete event simulation. The terms "process," "equipment," and "inventory" used in Table 2 are used simply to make this approach easy to understand in the context of a manufacturing process. For example, rather than generating a "reactor" resource at Step 3 in Table 2, the recipe could generate a "dirty reactor." A separate cleaning task could then consume this "dirty" reactor and generate a "clean reactor." A non-zero "inventory" of "dirty reactor" could serve as a rule for triggering this task. In a non-generic formulation, the model structure would need to be extended, using additional blocks and discrete items, to accommodate this cleaning activity.

3 IMPLEMENTATION – BATCH PLANT EXAMPLE

This section describes how the conceptual framework described in Section 2 can be implemented. To ensure clarity, the implementation is described for a specific example of a network, namely a batch plant. Application to other network types should be straightforward given the analogy that can be drawn between a batch plant and other network types (recall the discussion pertaining to Table 1). The description in this paper is limited to the simulation of the flow of materials in the plant to maintain the focus on the

generic simulation of a process network. In practice, simulation of a batch plant could also include additional elements such as production planning and scheduling, process failures and other unplanned events, and operator movements inside the plant, depending on the simulation objective. These additional model elements are out of scope of this paper.

3.1 Overview

There are two components to the implementation: data structure and simulation logic. These two components go hand-in-hand and must be designed together, as illustrated in this section. Although we implemented the approach in ExtendSim (Imagine That, Inc. 2011), the approach is general and can be implemented in any simulation software with comparable functionalities. Any software-specific features are clearly indicated to avoid confusion.

3.2 Data Structure

We start the model design with the data structure first, to ensure that the model is indeed generic and can be easily updated using input data alone. Once we identify what data to use to apply the model to different systems, designing a simulation logic compatible with the resulting data structure is relatively straightforward, compared to going the other way around. The scope of this section is limited to the data required to implement the generic framework for simulating a network (specifically the batch plant example).

We take advantage of the built-in database feature of ExtendSim. For ease of data management, we use separate databases to store input data tables pertaining to production capability and those pertaining to recipes. In addition, we use a database reserved for tracking the run-time data used in simulation logic execution. These databases are described in the balance of this subsection. Whenever appropriate, we set the parent-child relationships between data columns in various tables to enforce consistency throughout the model, particularly with entries that are considered as sets, e.g., process names, product names, inventory names, and equipment names.

It should be apparent from their structures that the tables provided in this subsection can be scaled to any number of processes, products, inventories, and resources. As shown in Section 3.3, the simulation logic can be written to allow input tables of arbitrary sizes.

3.2.1 Process Input Database

This database includes tables of data that specify the plant's production capability: process network, production sequence, batch size, batch time or rate, and resource availability. With these data in the input, it is possible to model very different plants by simply importing a different input database file.

Tables 3 and 4 show the data structure for process network and production sequence for the example illustrated in Figure 2. The row number corresponds to process ID. "Process Name" is a unique description assigned to each process. "Check if Batch Process" is set to true if the process capability is specified in terms of batch time or rate. "Previous Process ID" and "Next Process ID" specify processes that precede and follow the given process, respectively. In a relational database, parent-child relationships can be established between "Process Name" and "Previous/Next Process ID" so that the latter can be selected from among the former.

The remaining columns specify the rules used to control the timing of batch execution. "Start After Previous Done?" is set to true if the batch can start only after the previous process is finished. "Stop After Next Done?" is set to true if the batch can finish only when the next process is available. These criteria usually apply when the process lacks a buffer upstream and downstream, respectively. Otherwise, it is more convenient to trigger a batch start using the inventory in a buffer specified in "Start Based on Inventory ID." If the level set point is constant, then it can be entered in this table. If the set point value entered is zero, then the set point is calculated from the recipe table. If "Check if Inventory Replenishment"

is true, then a batch is initiated if the level of selected inventory is below set point. Otherwise, a batch is initiated if the level is above set point.

Table 3: Input data table – process network (example from Figure 2 top, without buffer)

ID	Process Name	Check if Batch Process	Previous Process ID	Next Process ID	Start After Previous Done?	Stop After Next Done?	Start Based on Inventory ID	Level Set Point	Check if Inventory Replenishment
1	A	X		2		X			
2	B	X	1		X				

Table 4: Input data table – process network (example from Figure 2 bottom, with buffer)

ID	Process Name	Check if Batch Process	Previous Process ID	Next Process ID	Start After Previous Done?	Stop After Next Done?	Start Based on Inventory ID	Level Set Point	Check if Inventory Replenishment
1	A	X		2			1	100	X
2	B	X	1				1	200	

For the simple example problem in Figure 2, the production sequence without the buffer is a straight forward "run the batch in A followed by B" (Table 3). When there is a buffer, batch execution depends on the buffer level or capacity. For this example, both A and B depend only on buffer C (inventory ID = 1 in Table 4), so the simple level set points can be used. Specifically, the example in Table 4 shows that process A runs whenever the level in C decreases to 100, and process B runs whenever the level in C increases to 200. If the last three columns are left blank, then the batch is started whenever all feeds specified in the recipe are available and finished whenever there is sufficient downstream capacity to accommodate all products specified in the recipe.

Table 5 shows the data structure for batch size and rate or time per process per product. Batch size and batch time/rate tables are used as look-up tables for these attributes, given the product ID (row number) and process ID (column number – 1). For the rate/time table, the value entered is used as batch time if "Check if Batch Process" is true in Table 3. Otherwise, it is used as rate.

Table 5: Input data table – batch size and rate or time

Product Name	Process 1	…	Process N

Tables 6 and 7 show the data structure for inventory and equipment, the two types of resources associated with production. The row number corresponds to inventory ID and equipment ID, respectively. In a relational database, a parent-child relationship can be established between "Inventory Description" (in Table 5) and "Start Based on Inventory ID" (in Table 4) so that the latter can be selected from among the former. Different resources for different network types can be set up in a similar manner.

Table 6: Input data table – inventory (example from Figure 2 bottom, with buffer)

Inventory Description	Minimum Level	Maximum Level	Initial Level
Buffer C	0	1000	500

Table 7: Input data table - equipment

Equipment Name	Number of Units

3.2.2 Recipe Input Database

This database contains a set of recipe tables, one per process per product. The structure of a recipe table is already provided in Table 2. Another table can be added to the database as a table index to specify which recipe table to use for which process and product. Table 8 is an example structure for the recipe table index. Product ID (row number) and Process ID (column number – 1) are the same as those established in the Process Input Database. In ExtendSim, the numerical ID associated with each table is used to specify the recipe tables in Table 8. The use of such table index greatly simplifies the recipe table maintenance.

Table 8: Recipe data table – table index

Product Name	Process 1	…	Process N

3.2.3 Runtime Database

This database contains a set of tables that are used to track various runtime data, which are used in simulation logic execution. Table 9 shows the types of information that need to be tracked during simulation to implement the generic framework for simulating a batch process. Additional data need to be tracked in simulations that include additional elements such as production planning and scheduling, process failures and other unplanned events, and operator movements inside the plant.

Table 9: Runtime data tables

Table	Data
Process Status	Current product ID, availability, batch status
Equipment Status	Number available
Inventory Status	Current level

3.3 Simulation Logic

The simulation logic described in this section is limited only to what is required to implement the generic framework in the context of a simulating the flow of materials in a batch plant. As mentioned earlier, other possible elements of a batch plant simulation are not included in the present discussion. Although some of these elements, such as process failures and operator movements can also modeled in a generic way, they are out of scope of this paper.

Figure 3 shows the high-level simulation logic for initializing the process unit items. First, N discrete items are created, where N is equal to the number of process steps in the plant (number of records in Table 3). Each process item is given a unique process ID, which is used to look up batch size and batch time or rate (Table 5). Any process that is given a batch size or batch time of zero in the input data is ignored. This feature allows certain parts of a plant to be turned on or off with ease during evaluation of multiple plant designs.

Figure 4 shows the high-level simulation logic for running a batch, which continues from Figure 3. Note that at this point a batch is associated with a specific process item. First, the product ID is assigned to the item. This step requires a separate logic for production planning and scheduling, which by its very nature is plant specific and therefore outside the scope of generic simulation that applies to multiple sys-

tems. For the present discussion, it suffices to assume that there is such logic that determines which product the plant must produce next. The process items then enter a queue in which they wait until the conditions required for process start are met. These conditions correspond to the rules described earlier in Section 2.1. Once the process item is released from this queue, the process status is updated to "running."

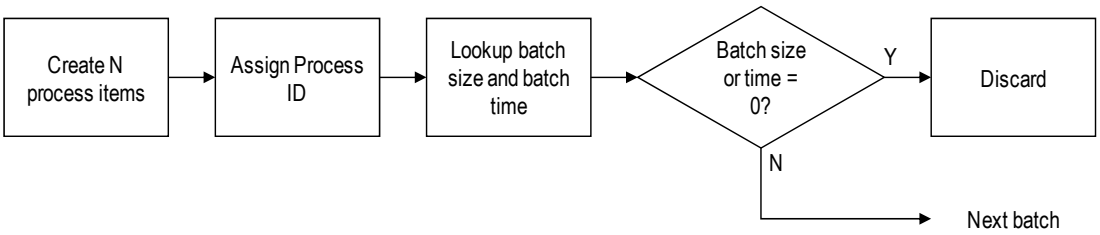

Figure 3: Simulation logic for initializing the process items

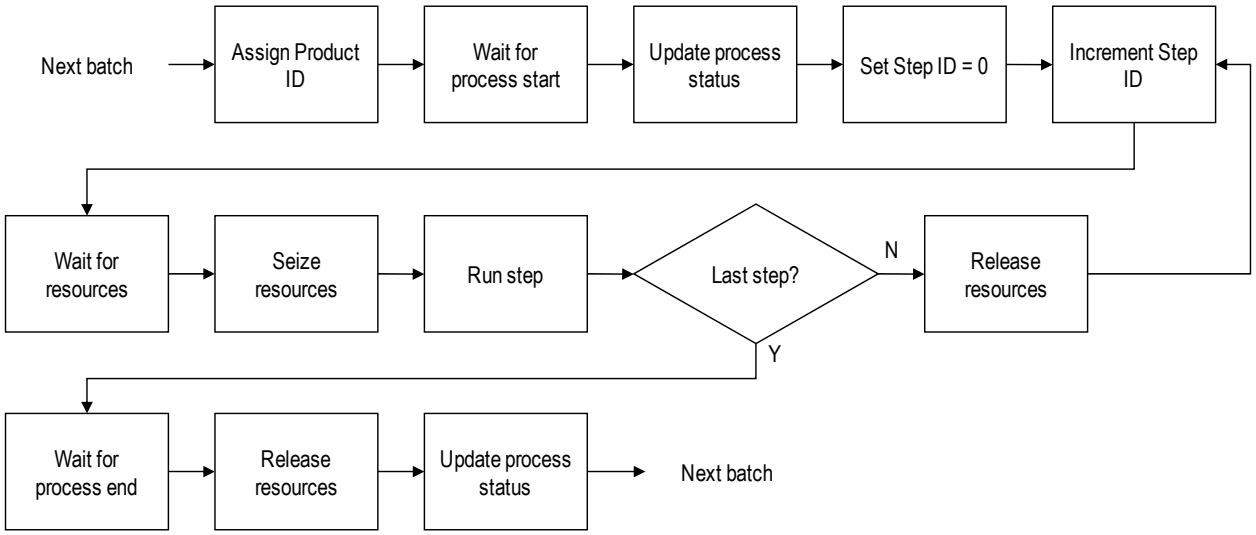

Figure 4: Simulation logic for the manufacturing process

The following pseudo-code is used to determine the item rank in the queue, which can be done using the Queue Equation block in ExtendSim. This code is executed for every process item in the queue, every time a new item enters or leaves the queue or the process status is updated.

1. Initialize the rank to default value of process ID
2. Look up the process start conditions for this process ID (Table 3)
3. Check the process status
4. Check the inventory levels
5. If this process is triggered by the previous process, stay in queue (set rank=null) until the previous process is finished
6. If this process is triggered by inventory:
    a. If inventory replenishment, stay in queue until the destination inventory level is at or below reorder set point
    b. If not inventory replenishment:
        i. If the level set point in Table 3 is zero, then calculate the set point from the batch size and unit ratio for the first inventory resource in the recipe (Table 2)
        ii. Stay in queue until the source inventory level is at or above the set point

The next part of the simulation logic involves stepping through a batch recipe, in a manner analogous to a real-life batch process control. At each step, the process waits for the availability of resources that are consumed, seizes those resources, and runs a queue similar to "wait for process start," in that the items are held (rank=null) until all required resources specified in the recipe are available. "Run step" is just a time delay for the duration of the step time. At the end of each step, the resources generated by this step are released. At the last step, however, the releasing of resources takes place only after the process end conditions are met. These conditions correspond to the rules described earlier in Section 2.1. Once the batch is completed, the process status is set to "not running."

4 APPLICATION EXAMPLE

One useful application of a generic plant model is evaluation of multiple plant designs. We applied this approach to capital project scoping, during which the capital project team considered different plant configurations. Figures 5-8 show several of the different configurations that were evaluated, using a single model. In addition, the number of trains per process changed multiple times during the course of this evaluation. The simulation logic described in Section 3 was slightly modified to allow any number of trains per process that can run independently of each other, so those changes could also be evaluated with the same model. In spite of the significant changes made to the system under evaluation, the use of a generic simulation model allowed much faster turnaround than would have been possible with a non-generic approach, in which separate models are constructed for different plant configurations.

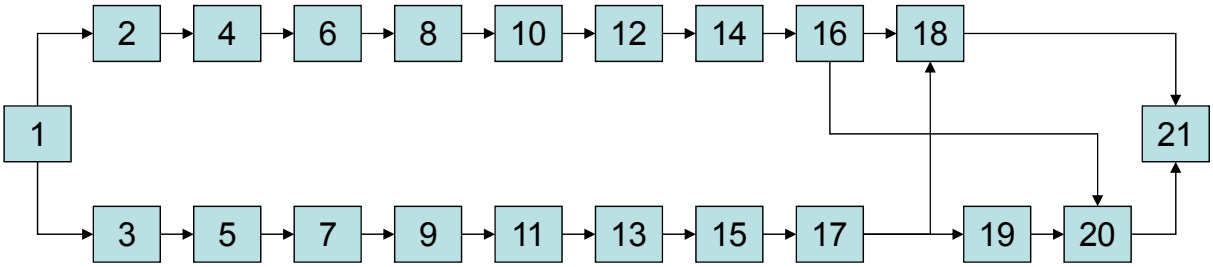

Figure 5: Implementation example – configuration 1

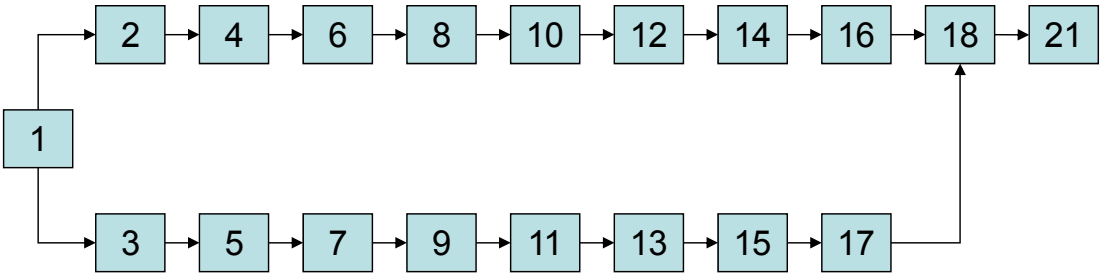

Figure 6: Implementation example – configuration 2

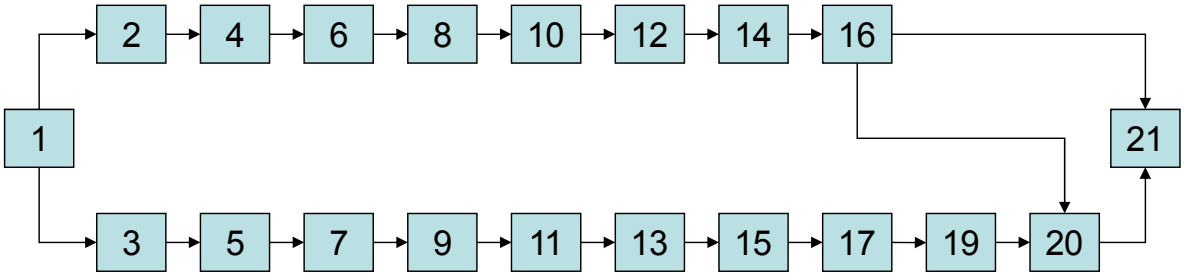

Figure 7: Implementation example – configuration 3

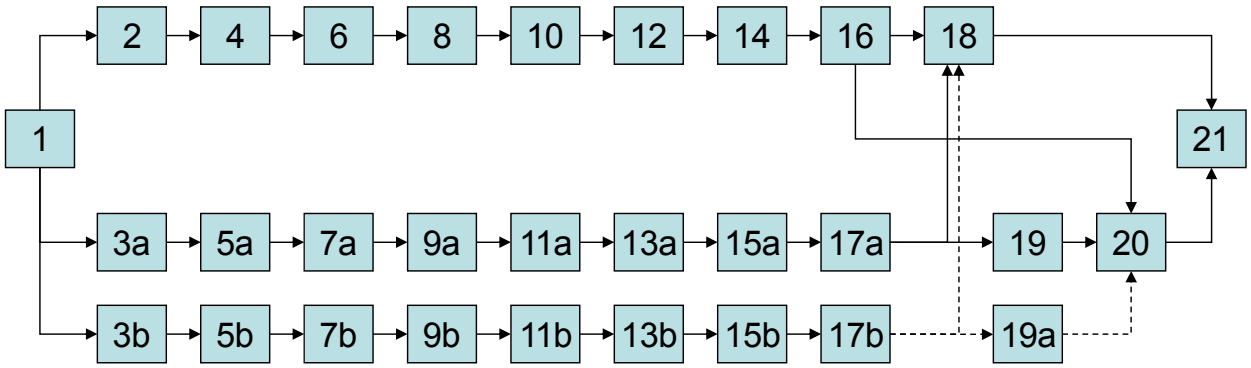

Figure 8: Implementation example – configuration 4

5 SUMMARY

We proposed a framework for building a generic discrete event simulation model of a network, which is characterized by flow of materials between interconnected nodes. The framework comprises two elements. The first element is the use of rule-based queue to control the timing of flow between various pairs of nodes. The network structure is not specified explicitly but emerges from the execution of these rules during simulation. The second element is the use of a recipe structure inspired from Resource-Task Network to model the timing of supply and demand of materials at each node, which allows for non-steady state operation of the nodes. We described conceptually the data structure and simulation logic that can be used to implement this framework in any simulation software. With this framework, it is possible to model a variety of system configurations with changes in input data alone, without any structural changes to the model.

Based on our experience, the main challenge in applying this approach is handling product-specific rules or constraints for a plant that produces a diverse portfolio of products. This challenge can be handled through appropriate design of recipe tables (i.e., Table 2 needs to be enhanced to meet the specific needs for a given system). Doing so, however, could require departure from an entirely generic framework. This issue remains an ongoing research topic for our group.

ACKNOWLEDGMENTS

The paper is based on the work supported by the Corporate Innovation Funding at The Dow Chemical Company.

REFERENCES

AspenTech. 2011. "Aspen Batch Process Developer – AspenTech." Accessed April 18. http://www.aspentech.com/products/aspen-batch-plus.aspx.

Bernal-Haro, L., C. Azzaro-Pantel, L. Pibouleau, and S. Domenech. 2002. "Multiobjective Batch Plant Design: A Two-Stage Methodology. 1. Development of a Design-Oriented Discrete-Event Simulation Model." *Industrial Engineering and Chemistry Research* 41:5727-5742.

Brown, N. A.. 2010. "Model Flexibility: Development of a Generic Data-Driven Simulation." In *Proceedings of the 2010 Winter Simulation Conference*, edited by B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, 1366-1375. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Bury, S. J., N. Akiya, and J. Ferrio. "Combining Simulation with RTN Optimization for Multi-product Chemical Processes." Case study presented at *2004 Winter Simulation Conference*, 5-8 December, Washington, DC.

Imagine That, Inc. 2011. "ExtendSim Simulation Software by Imagine That Inc." Accessed April 18. http://www.extendsim.com/.

McGinnis, L., and V. Ustun. 2009. "A Simple Example of SysML-Driven Simulation." In *Proceedings of the 2009 Winter Simulation Conference*, edited by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, 1703-1710. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Pantelides, C. C. 1994. "Unified Frameworks for Optimal Process Planning and Scheduling." In *Proceedings of the Second Conference on Foundations of Computer Aided Operations*, edited by D. W. T. Rippin, J. C. Hale, J. F. Daviss, 253-274. Austin, Texas: CACHE Publications.

Rockwell Automation. "Arena Simulation Software by Rockwell Automation." Accessed April 18, 2011. http://www.arenasimulation.com/

Selvaraj, R., X. Hua, and T. Hess. 2005. "Model-Centric Method and Apparatus for Dynamic Simulation, Estimation and Optimization." *U.S. Patent Application Publication* US 2005/0071137 A1.

Schönherr, O., and O. Rose. "First Steps Towards a General SysML Model for Discrete Processes in Production Systems." In *Proceedings of the 2009 Winter Simulation Conference*, edited by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, 1711-1718. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Wassick, J. M., and J. Ferrio. 2011. "Extending the Resource Task Network for Industrial Applications." *Computers & Chemical Engineering*. Available online 12 January 2011.

Winkel, M. L., L. C. Zullo, P. J. T. Verheijen, and C. C. Pantelides. 1995. "Modelling and Simulation of the Operation of an Industrial Batch Plant Using gPROMS." *Computers in Chemical Engineering* 19:S571-S576.

Yuan, Y., C. A. Dogan, and G. L. Viegelahn. 1993. "A Flexible Simulation Model Generator." *Computers and Industrial Engineering* 24:165-175.

AUTHOR BIOGRAPHIES

NAOKO AKIYA is a Research Scientist in the Process Optimization group of The Dow Chemical Company's Engineering Sciences organization within Core R&D, where she has worked on a variety of simulation and optimization projects to support data-driven decision making. She received her Ph.D. and M.S. from the University of Michigan, Ann Arbor, and B.S. from Rice University, all in Chemical Engineering. Her research interests include decision making under uncertainty, risk and reliability, and quality control. She is a certified Six Sigma Black Belt. Her email address is nakiya@dow.com.

SCOTT J. BURY is a Principal Research Scientist in the Process Optimization group of The Dow Chemical Company's Engineering Sciences organization within Core R&D. His research interests include process simulation & optimization of chemical processes using both continuous and discrete event technology and eliminating failure modes when implementing new process technology. He is a certified Six Sigma Black Belt. He is a member of INFORMS. His email address is sjbury@dow.com.

JOHN M. WASSICK is a Supply Chain Technology Consultant in the Supply Chain Technology Center of The Dow Chemical Company. His current research interests include large scale optimization and optimization under uncertainty, batch process optimization, multi-agent systems, and supply chain modeling and optimization. He holds a Ph.D. in Electrical Engineering from Michigan State University and he is a member of INFORMS. His email address is jwassick@dow.com.