Proceedings of the 2012 Winter Simulation Conference
C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A.M. Uhrmacher, eds

INTEGRATING DISCRETE-EVENT AND TIME-BASED MODELS WITH OPTIMIZATION FOR RESOURCE ALLOCATION

Teresa Hubscher-Younger
Pieter J. Mosterman
Seth DeLand
Omar Orqueda
Doug Eastman

MathWorks
3 Apple Hill Dr.
Natick, MA 01760, USA

ABSTRACT

Optimization's importance for technical systems' performance can hardly be overstated. Even small improvements can result in substantial cost, resources and time savings. A constructive approach to dynamic system optimization can formalize the optimization problem in a mathematical sense. The complexity of modern systems, however, often prohibits such formalization, especially when different modeling paradigms interact. Phenomena, such as parasitic effects, present additional complexity. This work employs a generative approach to optimization, where computational simulation of the problem space is combined with a computational optimization approach in the solution space. To address the multi-paradigm nature, simulation relies on a unifying semantic domain in the form of an abstract execution framework that can be made concrete. Because of the flexibility of the computational infrastructure, a highly configurable integrated environment is made available to the optimization expert. The overall approach is illustrated with a resource allocation problem, which combines continuous-time, discrete-event, and state-transition systems.

1 INTRODUCTION

The increasing exploitation of automation in technical systems combined with their consistently growing scale has rendered optimization of the operation of such systems an ever more valuable exercise. For example, with day rates as high as $100,000 and over, a small improvement of the trajectory that an ultra large crude carrier tanker sails comes with significant returns (Arnsdorf 2012, Nightingale 2010). As another example, reducing the carbon monoxide produced by a single vehicle has a multiplier effect that may be of the order of millions of cars sold (e.g., in 2008 alone Toyota sold as many as 351,007 Toyota Corolla (Toyota 2009)). As a consequence even small performance improvements are likely to have substantial impact in terms of cost, resources used, time spent, etc. Thus, optimization plays a role of key importance in the design of technical systems across the entire spectrum of industry and application domains.

To formally approach the optimization problem for technical systems, it is useful to formulate the problem in a mathematical sense, thereby opening up the opportunity to exploit the extensive body of mathematics literature on optimization. Indeed, once a mathematical formulation is available, optimization methods abound, including calculus-based methods, linear programming, integer programming, dynamic programming, calculus of variations, etc. (Nemhauser and Wolsey 1999, Rao 2009). Such optimi-

©2012 IEEE

zation often requires formulating the problem according to a mathematical structure that is then amenable to be combined with a mathematical solution approach.

While such an approach of *optimization by construction* often achieves excellent results, it involves a complicated and intricate step where the optimization of a technical system in a given sense must be transformed into a formulation amenable to the mathematical solution approach. This can prove to be a challenging task, in particular if the technical system is modeled such that it embodies a variety of models of computation. Moreover, many times the task is further complicated because the optimization is over properties that make it necessary to account for critical parameters that correspond to detailed phenomena. Such detailed phenomena are especially difficult to handle in a canonical mathematical formulation when they are difficult to abstract, for example because their effect is crucial in the combination of various different models of computation. Concretely, effects such as parasitic resistances and capacitances may be essential in properly modeling a system as a combination of continuous-time and discrete-event model parts. Yet such effects are likely to (prohibitively) complicate the optimization problem.

In addition to the combination of various models of computation in the problem space, the solution approach contributes its own model of computation, time scales, etc. In other words, the optimization approach itself often operates in a different model of computation than the problem space. For example, while a manufacturing operation may operate on a daily time scale, the optimization of the process may take a weekly or even monthly time scale and a model of computation different from the problem space may be chosen to (i) represent the solution approach and (ii) efficiently support the time scale of the solution approach.

With the different time scales between the optimization and the problem under optimization naturally comes the desideratum to change the problem under optimization over the course of operation and through various cycles of optimization. Facilitating dynamic problem adaptation necessitates automatically generating a new optimization problem, for example from an original model. However, while achievable in restricted cases, a more general such automation approach is a challenging undertaking.

Instead of a constructive approach the work presented in this paper employs a *generative approach* to mitigate many of the issues. As opposed to incorporating deep knowledge of the optimization problem in a formalization such that it can be exploited by the solution approach, a flexible and powerful interface between the problem and the solution space is defined. This interface enables the use of sophisticated simulation algorithms on the problem side and a broad spectrum of optimization methodologies on the solution side. Previous work based on a Knowledge Interface Broker (KIB) has demonstrated the value of such a generative approach where discrete event simulation is combined with Modeling Predictive Control (MPC) optimization algorithms that operate on a much larger time scale (Godding, Sarjoughian and Kempf 2007). While the knowledge interface broker introduces an elegant concept in solving the necessity for semantic adaptation between the problem and solution space, this paper presents a refinement based on an integrated architecture that eliminates the complexity of combining disjoint tools.

A main benefit of an integrated architecture is that it eliminates the complexity of having different tools with their various idiosyncratic nuances in the syntax, static and dynamic semantics of both interface as well as behavior. The challenge it poses in return is that it still requires the integration of problem and solution formulations, where these formulations often rely on different formalisms with different syntax and different semantics and where the problem and solution may operate on widely differing temporal scales. Moreover, within the problem formulation various different formalisms may be employed and so the overall challenge takes on a distinct multi-paradigm modeling character (Mosterman and Vangheluwe 2004).

This work addresses the multi -paradigm modeling challenge for optimization by developing an integrated optimization architecture with the objective of providing a unifying semantic domain (Harel and Rumpe 2000). Based on a well-defined abstract model of computation, elements in the architecture can register specific concrete refinements to implement various models of computation. An overall architecture framework around the abstract model of computation facilitates the integration of the various

refinements and provides a coherent and homogeneous optimization environment for multi-paradigm problems. As a case study, an optimization problem is formulated as model -predictive control of a batch process. Where the problem includes discrete-event and continuous -time models of computation, the solution approach relies on a control flow model of computation. The developed architecture then provides an integrated environment that enables direct experimentation with the different parts of the problem as well as the solution approach without changing development environments or refactoring interface brokers.

Section 2 of this paper first provides background to integrated optimization approach that is presented. Section 3 then discusses integration of different modeling paradigms. Section 4 introduces the case study model to be optimized, a chemical batch production model. Section 5 describes the optimization approach using MATLAB®.

2 BACKGROUND

The need for information aggregation and separation of input and output data for different modeling and optimization paradigms, based on syntactic and semantic simulation, as well as different time scales has proven to be a complex problem. The KIB approach with its specification of a set of nodes to handle the passing of information to the Discrete Event Simulation and Model Predictive Controller is useful for its ability to act as a transmitting and receiving layer for the knowledge that needs integration between the environments (Huang et al. 2009). The KIB provides a communication layer that allows the designers of the two different environments to act independently, but the integration is difficult to do during model generation.

As Fu (2002) argues, optimization in the solution space and discrete-event simulation in the problem space should be more integrated to take advantage of the fact that the two tasks are performed together. However, he also argues that integration of optimization into discrete-event simulation software can take a bad turn, when the user is supplied GUIs that hide the specific algorithms used and prevent the user from customizing the optimization routine. April, Glover, Kelly and Laguna (2003) argue a similar point when they describe the “metaheuristic black- box approach” in commercial software implementations of simulation optimization. Approaches often do not work well out-of-the-box, but need to be adapted to the specific characteristics of a problem.

The flexibility in MATLAB® (2012) and openness of its programming environment to optimization algorithms allows both integration with the simulation and the control for customization that is also desired for optimization routines that go beyond simple rules provided by the simulation tool.

Others have expressed the need for knowledge transfer between Model Predictive Control (Qin and Badgwell 2003) at one level of abstraction and Discrete Event System Specification (Zeigler and Sarjoughian 2000) at another. The challenge is that we must deal with the information exchange between enterprise system, which are consists of a base system, operative system, control and monitoring system and planning system (Mönch, Lendermann, McGinnis, and Shirrmann 2011).

We address some of these challenges in MATLAB through three types of integration:

1. Integration of multi-domain models using gateways
2. Integration through a base mathematical programming language
3. Parallel simulation of the hybrid model

3 INTEGRATION OF MULTI-DOMAIN MODELS

Integrating various modeling paradigms into a highly configurable and flexible environment requires support for combining continuous and discrete behavior, both in space and time. Such support comprises not only the ability for a user to express a notion in a particular paradigm but also the facilities for efficient computational algorithms operating on the paradigms. For example, a continuous-time model may be specified as a differential equation which is often solved computationally by applying numerical integration algorithms in a time stepping fashion (Mosterman, 2007). In contrast, a discrete-event model

may be specified as a discrete event system specification (DEVS) (Zeigler 1976) which is often solved by an event-driven approach, for example based on an event calendar.

In this work, continuous -time models are specified as time-based block diagrams, which are supported by the Simulink® (2012) graphical modeling environment. In addition to continuous-time, these time-based block diagrams support both discrete time with the notion of a sample rate as a first-class citizen of the modeling language. Discrete-event models are either specified as state transition diagrams, supported by the Stateflow® (2012) graphical modeling environment, or as entity flow diagrams, supported by the SimEvents® (2012) graphical modeling environment. In addition, untimed discrete event models are supported by the control flow approach of the MATLAB technical computing environment.

An integrated framework that supports all these various environments and their combinations is based on the S-Function Application Programming Interface (API) in Simulink. This API provides an abstract structure of a set of foundational execution operations. A subset that is of interest in the scope of this paper comprises

- An *output* operation to evaluate a combinational (Sander and Jantsch 2004) computation (*i.e.*, a functional computation without side effects such as modifying *state*).
- An *update* operation to evaluate a sequential (Sander and Jantsch 2004) computation (i.e., a computation that modifies the state of a dynamic system).
- A *derivatives* operation to obtain the derivative with respect to time of a model variable.
- A *function-call* operation to explicitly invoke an output operation, possibly combined with an update operation.

Function-calls are used with a type of subsystem or block that is invoked as a function by another block. In other words the function-call is a trigger that causes action in the component that the function call signal connects to.

An execution engine based on this framework then provides support for solving differential equations (implemented by making the output and derivative operations concrete), difference equations (implemented by making the output and update operations concrete), state transition behavior (implemented by making the function-call operations concrete), and entity flow behavior (implemented by making the function-call operations concrete and embedding an event calendar). Because of the shared framework, Simulink, Stateflow, and SimEvents modeling environments can be integrated to design models with the semantic heterogeneity that is often desired for complex optimization problems. Simulink and SimEvents allow discrete-event and time-based simulation models to be integrated with one another. The SimEvents engine works in conjunction with the different Simulink solvers to allow multi-domain modeling of complex systems.

In terms of model language elements, semantic clarity and precision is paramount. In this context, it is of particular importance to define the semantics of the interface between modeling paradigms (Hardebolle and Boulanger 2007). In this work, because of the integrated framework, the critical interface is between the imperative event-based entity flow diagrams of SimEvents and the declarative time-based block dia-grams of Simulink. To precisely define this interface, the imperative control that explicitly invokes an op-eration must be reconciled with a declarative formulation that is devoid of control over when an operation is invoked. A set of *gateway* blocks provide the user with language elements to specify the semantic adap-tation. Signals coming out of or driving the SimEvents model can be used in the time-based domain of the Simulink model via a component called the Gateway block. These Gateway blocks do signal conversion for use in the different domains. There are four types of Gateway blocks in SimEvents shown in Figure 1.

- *Timed to Event Signal* and *Event to Timed Signal* blocks adapt the event- driven imperative and the time- driven declarative environments of SimEvents and Simulink, respectively. The output signal of the *Timed to Event Signal* gateway block is identical to that of the input signal, except that now this signal can be used as an event-based input signal. The output signal of the *Event to Timed Signal* gateway block can be used for modeling time-based dynamics, and thus gives one value at any given time on the simulation clock. The output signal has a

sample time type of "fixed in minor step", in order to have sample time hits that are synchronized to the other time-based signals in the model.

- *Timed to Event Function-Call* and *Timed to Event Function-Call* blocks adapt the imperative state transition diagrams of Stateflow (with time derived from the overall execution engine) and the imperative entity flow diagrams of SimEvents (with time derived from the embedded event calendar). The *Event to Timed Function-Call* block converts the function-call from the discrete-event domain into one that can be used in a time-based domain, whereas the *Timed to Event Function-Call* converts the signal in the other direction.

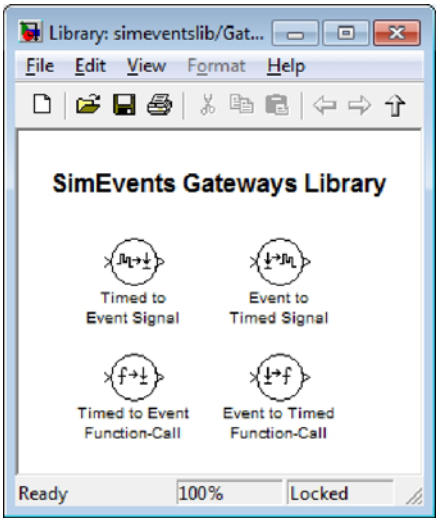

Figure 1: Gateway blocks that are part of the SimEvents Gateways Library help transform signals from a discrete-event system to work in a time-based system and vice versa.

These gateway blocks clearly define where system boundaries and delineation of the two types of systems are, as well as allows for the passing of information from one type of system to another. We will demonstrate the use of these blocks in the example that follows.

4 THE PROBLEM TO BE OPTIMIZED—CHEMICAL BATCH PRODUCTION

The model the authors created to demonstrate how we support integration is a hybrid, discrete- event and time-based model that simulates a chemical batch production process with shared resources that processes two types of orders. There are three main components to this model (Figure 2): the order system, production system and process analysis. The order system generates orders with different demands on the production resources. The production system the loop in the middle, is assigning batch reactors to a particular order and processing this order. The process analysis component displays how the chemical production process resources were used over time.

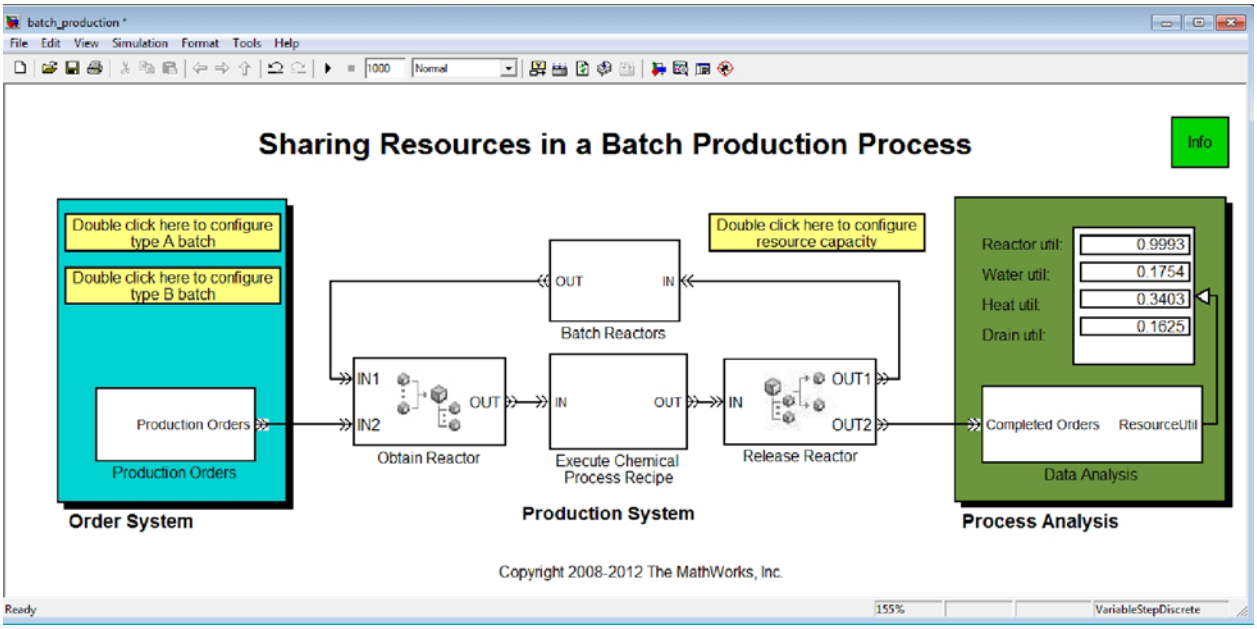

Figure 2: The chemical batch production model has three main components at the top level: the order system, the production system and the process analysis component.

The Production Orders subsystem (Figure 3) simulates the generation and backlog of two types of production orders, an A and B batch and then holds the order in a queue until a reactor is ready. A situation where this might be applicable is where the A batch being a weaker strength of a particular pharmaceutical and B being a stronger one. They each would require different amounts of time of the shared resources to be available.

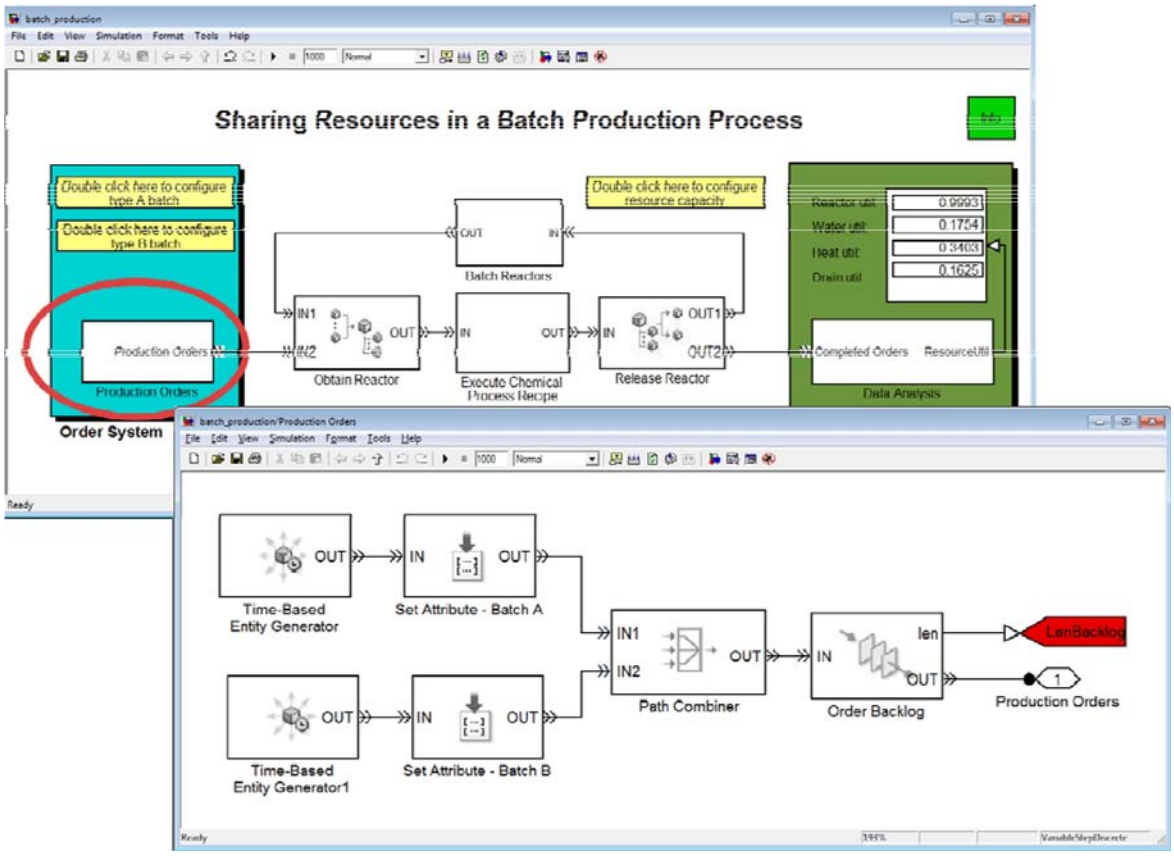

Figure 3: The discrete-event subsystem of the Production Orders component that creates two different batches of orders.

The Batch Reactors subsystem (Figure 4) models batch reactors, one of the main shared resources in this production process. One reactor is used to process one production order or batch. The Time-Based Function-Call Generator and Event-Based Entity Generator generate the batch reactors at the beginning of the simulation. The FIFO Queue block holds the reactors until they are needed. The Path Combiner block adds back into the process batch reactors that have been used and are ready to be used for another order.

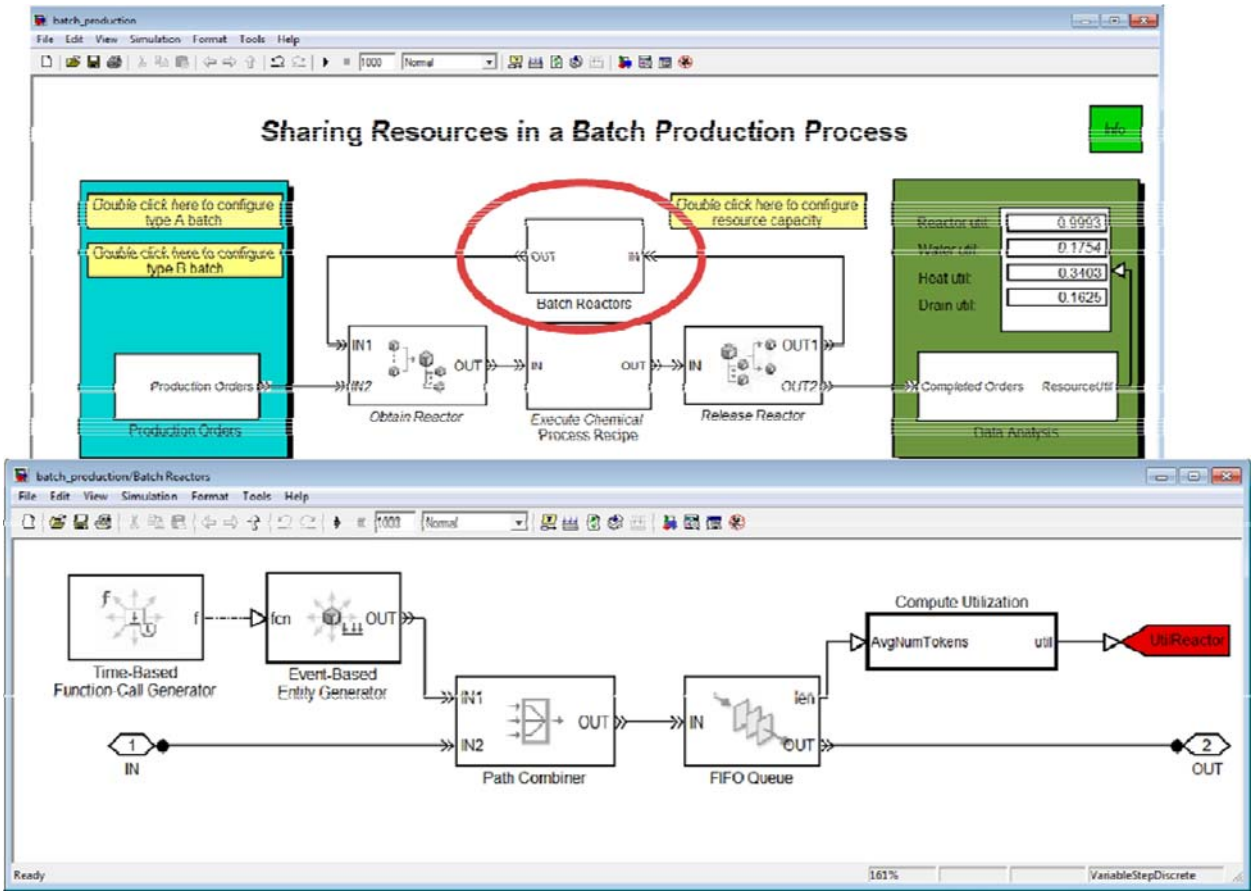

Figure 4: The Batch Reactors subsystem creates batch reactors at the beginning of the process, which are reused during simulation.

In the main model, once a particular production order is received, the block labeled Obtain Reactor requests a reactor from the set of batch reactors held in the Batch Reactors subsystem. This begins the chemical process. Inside the Chemical Process Recipe, there are a number of manufacturing steps, such as adding water, heating the mixture, adding color and particles, agitation and draining. (Figure 5). Once the Chemical Process Recipe is complete, then the batch reactor is released to be used for another order.

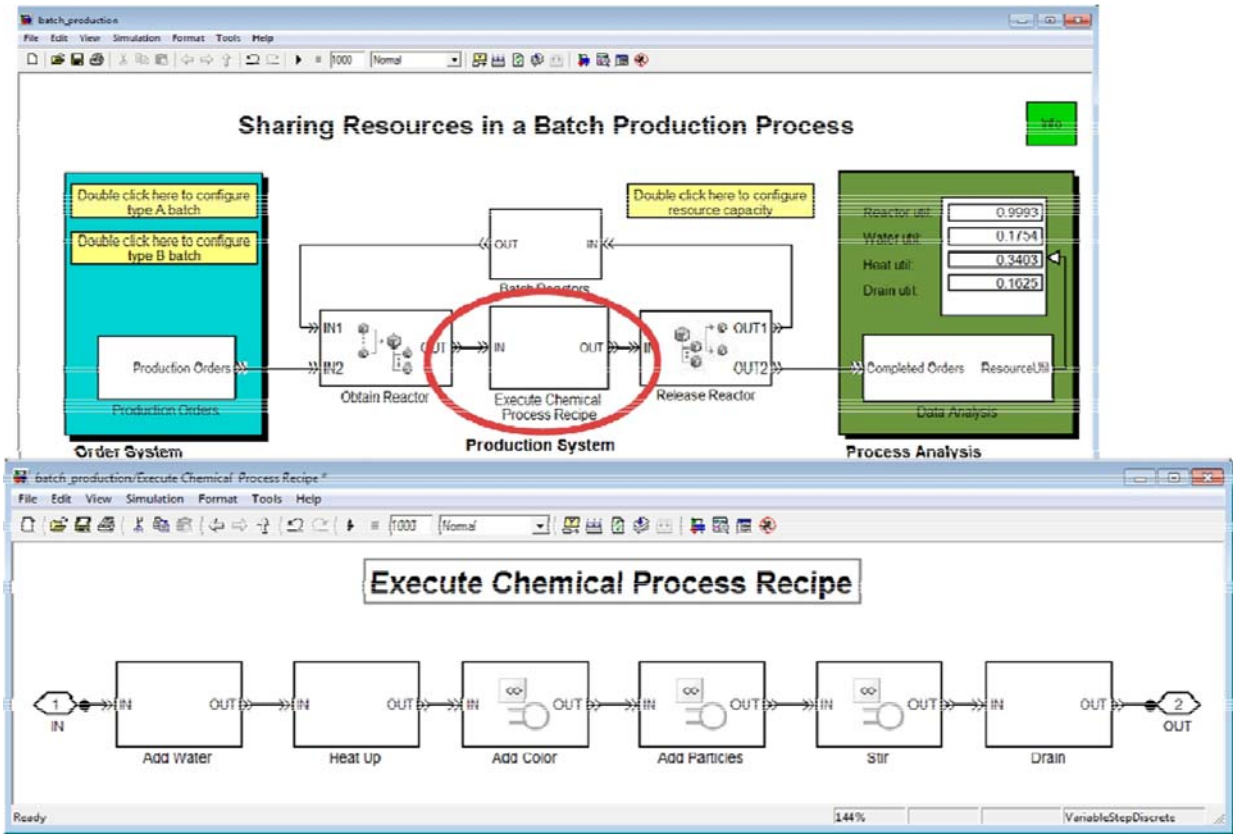

Figure 5: The Execute Chemical Process Recipe subsystem has the major steps of the batch reactor.

One of the subsystems, the Heat Up system, has a continuous time controller in the loop. This is where we see the usefulness of the gateway blocks. The Chemical Reactor Controller is a subsystem containing a continuous time MPC controller inside it, so it has two gateway blocks. One on the signal coming out of the discrete-event system which sets the feed concentration, and another gateway block that is used for enabling further orders from moving through the system based on the state of the controller (Figure 6).

Hubscher-Younger, T., Mosterman, P.J., DeLand, S. Orqueda, O., & Eastman, D. (2012). Integrating discrete-event and time-based models with optimization for resource allocation<http://ieeexplore.ieee.org/xpl/login.jsp?reload=true&tp=&arnumber=6465258&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6465258>. Proceedings of the 2012 Winter Simulation Conference.