Proceedings of the 2016 Winter Simulation Conference
T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S. E. Chick, eds.

SIMULATION OPTIMIZATION IN DISCRETE EVENT LOGISTICS SYSTEMS: THE CHALLENGE OF OPERATIONAL CONTROL

Timothy Sprock
Leon F. McGinnis

School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332

ABSTRACT

Simulation optimization tools have the potential to provide an unprecedented level of support for the design and execution of operational control in Discrete Event Logistics Systems (DELS). While much of the simulation optimization literature has focused on developing and exploiting integration and syntactical interoperability between simulation and optimization tools, maximizing the effectiveness of these tools to support the design and execution of control behavior requires an even greater degree of interoperability than the current state of the art. In this paper, we propose a modeling methodology for operational control decision-making that can improve the interoperability between these two analysis methods and their associated tools in the context of DELS control. This methodology establishes a standard definition of operational control for both simulation and optimization methods and defines a mapping between decision variables (optimization) and execution mechanisms (simulation/ base system). The goal is a standard for creating conforming simulation and optimization tools that are capable of meeting the functional needs of operational control decision making in DELS.

1 INTRODUCTION

Increasing scale, complexity, and connectedness have accentuated a need to improve decision-making support in Discrete Event Logistics Systems (DELS), a class of dynamic systems that create value by transforming discrete flows through operations performed by a network of interconnected subsystems (Mönch et al. 2011). These systems share a common abstraction: set of products flowing through a set of processes being executed by resources configured in a facility. The DELS domain includes supply chains, manufacturing systems, transportation networks, warehouses, and health care delivery systems.

One approach to improving decision-making in DELS is to make the use of simulation optimization methods more accessible to DELS decision makers. While improving optimization and simulation methods themselves would contribute to this goal, it is not enough. A more important improvement would be to achieve DELS-specific semantic as well as syntactic interoperability. This would support making simulation optimization methods a routine analysis, as well as addressing the difficult of interfacing individual simulation and optimization tools, generating and evaluating large-scale simulations, and interfacing with data sources and execution systems (Sprock and McGinnis 2015).

As we have argued elsewhere, a formal system model enables many of these requirements to be met (Sprock and McGinnis 2015). However, while good progress is being made on modeling the structure and behavior of DELS, there is an unmet requirement for a specification and conforming analysis methods for operational control. The lack of a formal model of operational control makes defining and searching the control policy space difficult, which often to leads to simplifications such as exploring a pre-defined set of control policies, such as FIFO in queues or round-robin routing. An agreed upon and explicit definition that integrates both the decision variables in the optimization model and the control execution mechanisms in simulation would provide a significant step forward in facilitating interoperability between the two analysis methods.

This paper provides a brief overview of simulation optimization methods for the DELS domain, then identifies the need for a model of operational control that can provide the necessary infrastructure for achieving interoperability between simulation and optimization methods and tools. We then address the modeling gap by proposing a model of operational control that includes the fundamental set of DELS operational control activities, as well as the necessary infrastructure to specify and execute control behavior. Finally, we discuss a wide range of research and implementation challenges that must be addressed to make this a usable and routine methodology for DELS decision support.

2 SIMULATION OPTIMIZATION METHODS IN DELS

Simulation optimization methods, which in their most general form aim to efficiently use a simulation model as an evaluation function integrated with search algorithms, are a promising approach for providing decision support for the DELS domain. In fact, simulation optimization methods already have been applied to solve a number of problems in this domain; for a representative sampling see, e.g. supply chains (Truong and Azadivar 2005), manufacturing (Kapuscinski and Tayur 1999), transportation (Cheng and Duran 2004), warehouses (Rosenblatt et al. 1993), and healthcare (Ahmed and Alkhamis 2009).

For DELS, domain-specific simulation optimization methods already have been developed that exploit several common characteristics of the domain to provide narrowly-scoped and well-structured search neighborhoods, such as focusing on the network structure, resource investment and behavior, or operational control policies. In genetic algorithms, e.g. the domain specific structure can be exploited through specialized encoding schemes, initialization procedures, and local search operators, see e.g. (Azadivar and Tompkins 1999, Zhou et al. 2002, Syarif et al. 2002, Ding et al. 2009, Costa et al. 2010). In Tabu methods, domain specific search neighborhoods can be integrated into the local search procedure, see e.g. (Alabas et al. 2002, Yang et al. 2004). Finally, knowledge-based optimization methods incorporate learning modules that integrate domain specific information to guide the optimization search process (Huyet and Paris 2004). However, all approaches reported to date have depended upon problem instance-specific optimization and simulation formulations.

2.1 Simulation Optimization Interoperability

One of the challenges in developing plug-and-play simulation optimization tools is the difficulty of bridging between the two generic tools because the syntax of the simulation and optimization methods and their tools are fundamentally different. While optimization methods can use more general or abstract representations which maximize the method's applicability to a broader class of problems, simulation methods, and the resulting models, almost always operate on concrete semantics of a particular problem domain. Bridging between the abstract semantics of optimization and the concrete semantics of simulation is necessary to achieve interoperability of their respective solvers. The SimOpt library (Pasupathy and Henderson 2011) and the related research have focused on developing syntactical interoperability, which has allowed the development of more generic optimization methods, see e.g. COMPASS (Xu et al. 2010) or evolutionary algorithms (Zhang and Li 2007).

As is often the case with integrating COTS tools, the mapping between tools with different syntaxes, i.e. integrating a generic optimization tool suite with a new simulation tool, is done once for each pair of tools (Fu et al. 2014). Then a second mapping is required for each particular problem instance to align their semantics. However, the burden of mapping and translating the output of the optimization model falls on the interface to the simulation tool; e.g. translating a semantic free array of values from the optimization solver into a network structure, e.g. supplier selection, or resource sizing and selection, e.g. parallel machine counts at each workstation, or the selection of a particular control rule from a pre-defined collection. The mapping from a simulation parameter to an optimization variable is done by hand for each system instance by "promoting" a particular tunable simulation value, i.e. for a given simulation model, manually selecting the set of simulation parameters that can be modified during the optimization process.

The interface to any particular optimization solver is, in general, quite straightforward because there is a trivial mapping from the abstract semantics of the analysis model to the concrete semantics of the associated solver. The same is not true on the simulation side, in particular, because there is no generic abstract semantics for a simulation analysis model –there is only the concrete semantics of a particular simulation language (and its associated solver). The main consequence of this asymmetry is that it is relatively easy to change optimization solvers, but quite difficult to change simulation solvers. Moreover, because the semantics and syntax of various simulation solvers are so different from one another, it is practically impossible to have a generic, reusable abstract model of system behavior and control. A common modeling language would enable a natural mapping between the abstract mathematical description of solution algorithms and the system models that provide the necessary context and data, and is critical to bridging the gap between these analysis models and practical implementations of their corresponding tools.

2.2 Bridging Between Simulation and Optimization Methods Using a System Model

Several modeling methodologies have been proposed that use a system model independent of the simulation model to enable more generic interoperability between the simulation and optimization tools. Jeong (2000) develops a conceptual framework rooted in the IDEF standard for optimizing a simulation-based scheduling system. This conceptual framework suggests several requirements for implementation including generating DES from a production database, interfacing simulation and optimization tools, and implementing a rule-based controller for scheduling policies. Sivakumar (1999) uses UML to model the system and generate the associated interfaces between the simulation, optimization, and data sources.

Extending these approaches, the simulation and optimization methods can be mediated with a formal system model, where the simulation model is a view generated from the complete system model and the optimization model must translate its output into the language of the system model (Sprock and McGinnis 2015). Generating and updating a simulation model from a single model of the system allows the optimization results to be used to modify the structure of the simulation model, e.g. network optimization or building state machine representations of resource or control behavior. Also, it allows multiple views, or simulation models of varying resolution and fidelity, to be generated consistently from a single representation of the system.

While facilitating interoperability between simulation and optimization tools with a system model that provides a semantic description of the structure and behavior of the DELS has been previously addressed in (Sprock and McGinnis 2015), a canonical model of control for the DELS domain, especially one that can be implemented in discrete event simulation tools, remains elusive. That is, there does not exist a universally accepted model of the control decisions that are routinely required to guide the operational behavior of a DELS or a method to implement those decisions in tools without resorting to ad-hoc code. For example, control specifications are notably absent from standards such as SCOR or CMSD (Leong, Lee, and Riddick 2006, SCOR 2012). Even within ISA-95, the Level 3 specification for manufacturing operations management does not provide complete guidelines on control decision making (Scholten 2007).

3 MODELING OPERATIONAL CONTROL

Implementing supervisory control requires some external decision process (control) to be applied to the resources executing processes in order to guide the evolution of the system's behavior. However, for DELS, there is no formal, canonical specification or design methodology for this external control process. Many of the formalisms used to model and design control, e.g. controlled Markov chains, controlled petri nets, or extended petri nets, require some external function to be applied to the system to help it evolve. There is not a definitive way to construct these functions as they "follow no general rules, but depend on the logic of the specific problem at hand and the desired system behavior" (Valavanis 1990). A standard representation of the set of operational control problems would enable a uniform interface to solution tools thereby creating opportunities for interoperable, or plug-and-play, analysis tools. The operational control model that we propose addresses this issue by proposing three essential control functionalities: 1) an analysis-agnostic and functional definition of the control mechanisms themselves, 2) a mapping between decision-variables (optimization) and execution mechanisms (system/simulation), and 3) the representation of the operational control decisions or the interface between the simulation/optimization tools.

3.1 Separation of Plant and Control Specifications

Modeling operational control mechanisms first requires the separation of plant and control within the system specification, as well as conforming simulation models. Consider the contrasting approaches for designing the control logic of discrete event systems discussed in (Holloway et al. 1997). In the controlled behavior approach, the open-loop behavior of the Petri Net is modified until it exhibits the desired closed-loop behavior; however, then the control logic must be extracted from the Petri Net to produce the required code. This is typical in the status quo where the control behavior in simulation languages is embedded in the blocks that execute the behavior, e.g. queuing discipline is modeled as a property of the queue block. This is quite different the logic controller and control theoretic approaches, the design method is "[focused] on the direct design and implementation of the controller for the plant" and there is a "clear distinction between the plant and the controller" (Holloway et al. 1997).

This separation between plant and control is essential for developing operational control design methods that specify the control behavior separately from its execution by a control actuator in the plant. However in the DELS domain, there remains a need for discrete event simulation language support for specifying control rules and methods in such a way that they can provide a pathway from design and analysis of controllers to prototyping and testing to deployment. This problem already has been addressed in mechatronic or other engineered systems, where the control logic can be designed in Simulink and then ported directly into a real controller deployed within the actual system being designed (Grega 1999, Piltan et al. 2012). While it appears that the SimEvents simulation tool (Mathworks 2015), which is closely related to Simulink and allows for the control behavior to be specified as a finite-state machine, is promising approach, there remains a need for more prevalent control design methodologies and tools that explicit separate the designed control behavior from the system specification (see Figure 4 in section 3.4.3 for further details).

3.2 Extant Research on Control Policy Optimization in DELS

Control policy optimization methods are quite diverse, from static design methods to dynamic, online scheduling methods, as is their output, from rules to artificial neural networks (Kusiak and Chen 1988, Davis et al. 1992, Kim and Kim 1994, Lee et al. 1997, Pierreval and Mebarki 1997, Huyet and Paris 2004, Mouelhi-Chibani and Pierreval 2010, Nie et al. 2010, Kapanoglu and Alikalfa 2011, Zhang and Rose 2014, Branke et al. 2016). There is a significant amount of research conducted in the area of simulation-based scheduling and dispatch for semiconductor and flexible job shop classes of problems. However, there is limited research on the other facets of operational control decision making or applying these methods to other classes of DELS.

Complex but static priority rules seek to optimally sequence tasks for a broad range of system states, see e.g. the use of hyper-heuristics to produce an optimal priority dispatch rule (Branke et al. 2016) or gene expression programming to optimize a priority rule expressed as a grammar (Nie et al. 2010).

Ranking and selection control methods enumerate and simulate n control policies, and then select the best performing rule to implement (Davis et al. 1992, Kim and Kim 1994). The controller then waits until the expected performance or schedule deviates significantly from the planned trajectory, and repeats the process. In a contemporary treatment of this method, Zhang and Rose (2014) evaluate the dispatching problem as an online sequential decision problem where the simulation-based method also simulates future decision points, where the dispatching rule may be changed again.

Dynamic rule selection methods in the form of decision trees or rules can be configured offline and incorporated into the controllers decision making, see, e.g. (Lee et al. 1997, Pierreval and Mebarki 1997, Huyet and Paris 2004, Kapanoglu and Alikalfa 2011). Lee et al. (1997) apply genetic algorithms and machine learning techniques to contruct decision trees that select the rules to release of jobs into shop floor. Genetic algorithms can also be applied to configure ‘if-then’ priority rules based on the state of the system (Kapanoglu and Alikalfa 2011). More general simulation optimization methods can be used to determine the numerical thresholds for dynamically selecting a rule from a collection of predetermined priority rules (Pierreval and Mebarki 1997).

Artificial neural networks (ANN) also can be used to support the selection of the dispatch rules dynamically to meet changing goals or new requirements, and can be trained off-line using simulation-optimization experiments (Mouelhi-Chibani and Pierreval 2010). Furthermore, a rule-based representation can be extracted from a trained ANN (Andrews et al. 1995). Operations research techniques can be used to design rule-based systems, and various architectures consider operations research analysis models and solutions tool as part of their structure, e.g. tandem expert systems (Kusiak and Chen 1988).

In the status quo, there is simply no broad methodological and tool support for designing and executing complex control behaviors for DELS. Applying these methodologies for designing and executing dynamic control methods to the broader DELS domain requires a more abstract model of operational control.

3.3 Fundamental Control Questions

Designing the operational control mechanisms for DELS first requires an explicit description, specification, or model of the complete set of operational control problems that any controller must be able to solve in order to be effective in managing the behavior of the system. This section describes a fundamental set of control questions that can be extracted from a relatively small collection of distinct control problems that are addressed in the literature.

Deriving a canonical set of control questions can be viewed from the perspective of defining a comprehensive functional specification of all the control mechanisms that a controller needs to be able to provide. We believe the canonical set of questions is: (1) “Should a task be served?” (admission); (2) if so, then “when should the task be serviced?” (sequencing); and, (3) “by which resource?” (assignment); (4) finally, “where should the task be sent after it’s complete?” (routing); (5) as well as the resource-related “when does the state of the resource need to be changed?” (resource state change). The associated decisions are illustrated in Figure 1.

Figure 1 also captures all of the decision points, or actuators, that each job interacts with or can be influenced by as it flows through the DELS. These actuators must interface with the controller (dashed lines) that provides control instructions which are then executed by the actuator in the base system.

While many existing analysis models address one or more of these questions jointly in order to achieve better quality solutions, a model of the atomic control decisions is necessary for specifying and designing the underlying execution mechanism. That is, a complete model of the individual control mechanisms provides a concise and reusable mapping of control decisions to execution mechanisms. For example, the scheduling decision is executed by two different control actuators, the one that sequences jobs and the one that then routes/assigns them to the target resource and “seizes” the resource from the pool. A complete discussion and argument for the completeness of this collection of control behaviors for the DELS domain is beyond the scope of this paper, but interested readers can see (Sprock 2015).

3.4 Proposed Control Model

The control questions help to identify the interoperability mechanism by linking the decision variables in the optimization model to the execution mechanisms in the simulation model (or “real” base system). For

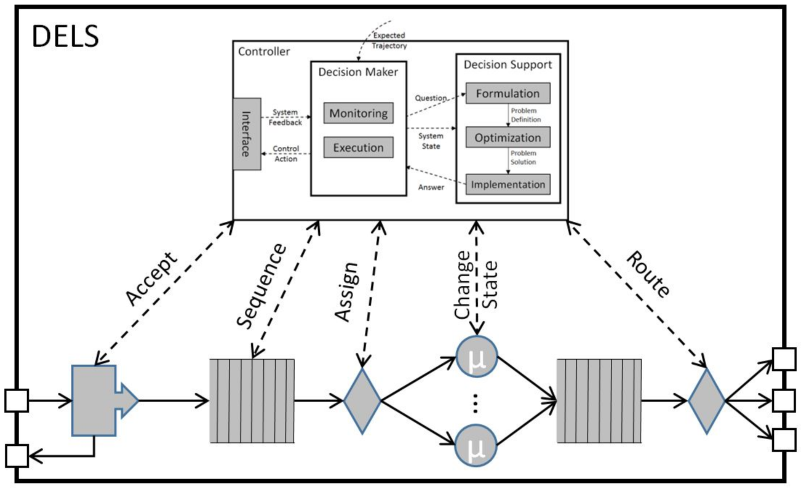


Figure 1: Stylized DELS process map with control 'actuators'.

each control question, the proposed control model defines a consistent interface for formulating control analysis models (section 3.4.1), uses rules and state machines to provide an interoperable definition of the control behavior (section 3.4.2), and provides a uniform execution mechanism and actuator in the plant (section 3.4.3).

Throughout this section, the admission control mechanism for a manufacturing workstation is used to illustrate an implementation of the control model that pairs a control execution mechanism (the actuator) with a control decision mechanism (FSM). The simulation tool SimEvents provides a library of primitive simulation components, which allows an explicit modeling of the separation of plant and control and the actuator mechanisms in the system. The control behavior is implemented using the state machine model provided in the StateFlow library.

3.4.1 Formulating a Control Analysis Model

Formulating an analysis model to answer a particular instance of a control question should rely on a reusable and extensible mapping from the system model to an analysis model that is centered on the abstract question definition. This reusable mapping is important because for each control question, there may be several analysis methods available to provide an answer, each with different solution quality and run-time performance guarantees. To answer a particular control question, the controller should rely on the context of the question and the system model to construct a particular answering analysis model.

A uniform interface to analysis models and their corresponding tools enables each controller to access a wide variety of solution methods for each control question. The idea of a uniform interface is embodied by the strategy pattern (Gamma et al. 1994), which defines a family of algorithms, encapsulates each one, and makes them interchangeable. The formulation component is configured with an abstract strategy class for each control question to formulate the corresponding analysis models, where the target question defines the decision variables, the motivation defines the objective function, and the system model provides the instance data, including the constraints associated with each task and resource. Each concrete strategy uses this information in different ways to construct different analysis models, but the uniform interface allows the details of this construction process to be encapsulated, or hidden, in the class's behavior definition.

The example in Figure 2 illustrates the strategy pattern approach. Two possible admission strategies are given an incoming task as input and determine whether to admit the task or not. For illustrative purposes, these strategies are simple and only consider the amount of WIP in the system (*simpleCapacity*) and the priority of the incoming task (*capacitatedPriority*).

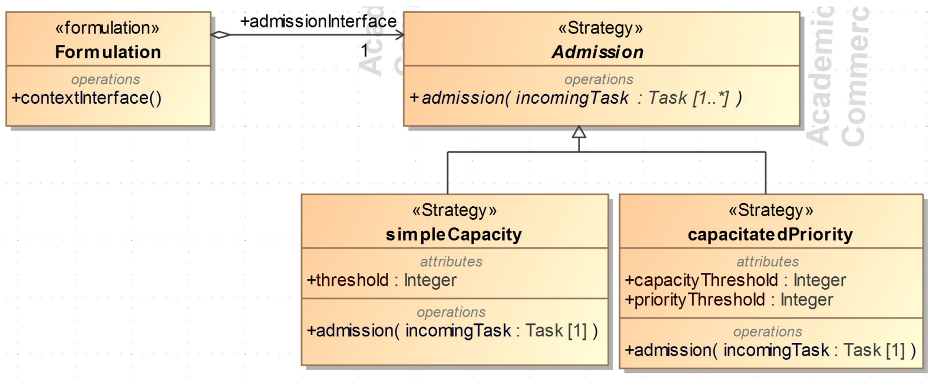


Figure 2: The strategy pattern defines a consistent interface invoking control analysis methods.

3.4.2 Control Behavior Formalism — The Importance of Rules

Formulating an analysis model that can be shared with a solver typically requires translating the problem into mathematical syntax, thereby stripping any domain knowledge or semantics from the problem. As a consequence, the solver solution provides an answer to the control question in the same semantic-free abstraction. Therefore, there remains a need for a formal specification of the output of optimization solvers that adequately expresses the optimal control decisions to be executed. Rules in the form of condition/action (production rules), event/condition/action (ECA rules), or pattern/action (complex rules) are knowledge representation methods capable of capturing control behavior, see, e.g. (Iassinovski et al. 2008, Shirazi et al. 2010) for applications in the DELS domain. Finite state machines (FSM) and production rule systems (PRS) are capable of organizing these rules and providing an execution mechanism to decide when and which rules to activate. In Figure 3, the output of the admission strategy discussed in section 3.4.1 is captured as a FSM that can be translated into simulation code.

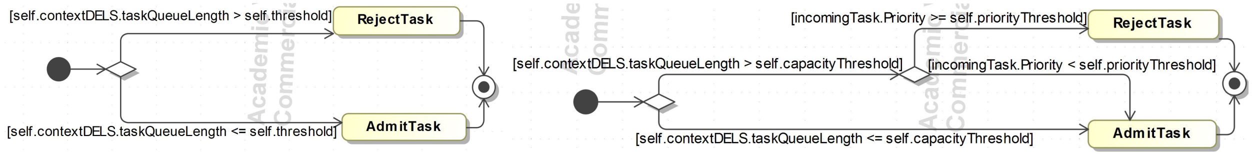

[self.contextDELS.taskQueueLength > self.threshold]
RejectTask
[self.contextDELS.taskQueueLength <= self.threshold]
AdmitTask
[incomingTask.Priority >= self.priorityThreshold]
RejectTask
[self.contextDELS.taskQueueLength > self.capacityThreshold]
[incomingTask.Priority < self.priorityThreshold]
[self.contextDELS.taskQueueLength <= self.capacityThreshold]
AdmitTask

Figure 3: The finite state machine formalism corresponding to the two strategies in Figure 2.

The result is an explicit mechanism for implementing the output of analysis tools by first restoring the semantic content to the analysis solution and transforming that solution into formal rules, which then can be executed using a finite state machine, production rule system, or a comparable tool. This is an important requirement for interoperability between analysis tools and mechanisms to execute control actions in the system, especially with a wide variety of control behavior representations used in the literature (section 3.2).

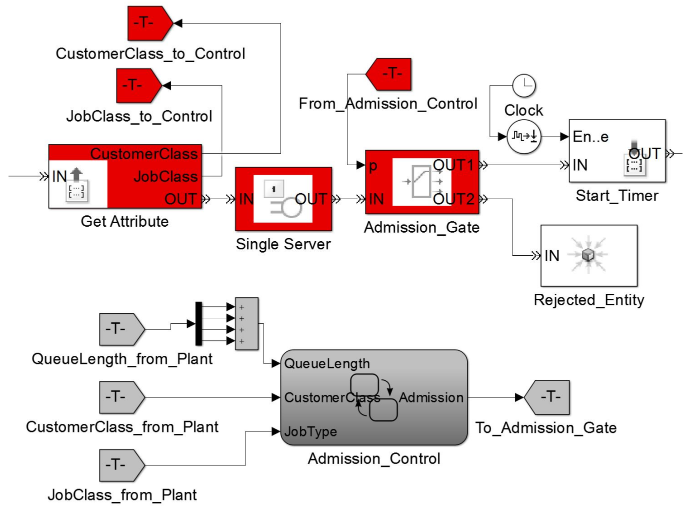
3.4.3 Control Execution

While finite-state machines provide a formalism to model the control behavior, a uniform, well-defined execution mechanism is the last component required to facilitate interoperability because it specifies the expected behavior from the system, or simulation model. The execution behavior consists of three components: (1) a computational rule (which provides the abstract specification of the execution behavior), (2) an actuator that implements the execution behavior, and (3) a *callBehavior* method that the controller uses to invoke the actuator's behavior.

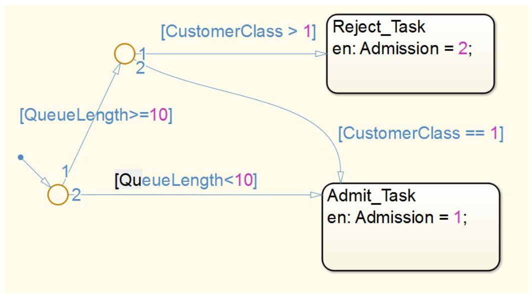
For the admission control behavior, the computational rule specifies that the *AdmitTask* behavior adds the incoming task (or job) to the system's task-set (queue).

$$\text{AdmitTask(task)} =_{Def} \text{System.TaskSet} \cup \{task\}$$

This execution behavior is implemented by the *Admission Gate* actuator block in Figure 4. In this SimEvents example, the *Admission Gate* is implemented as an admission routing block so rather than invoke the callbehavior *AdmitTask*, the *AdmitTask* behavior routs incoming jobs into the system which adds it to the system's queue. In Figure 4(a), there is clear separation of plant (red blocks) and control (gray blocks) in the system model with a well defined interface for exchanging data and control (the -T- blocks). The StateFlow block *Admission-Control* contains the control behavior specified as a FSM (Figure 4(b)), which implements the *capacitatedPriority* behavior from Figure 3.



(a) The simulation model of the admission mechanism.



(b) The finite state machine captures the admission control behavior.

Figure 4: The simulation model reflects clear separation of plant (red blocks) and control (gray blocks). The admission control mechanism is implemented as a admission routing block and a finite state machine.

4 RESEARCH AND IMPLEMENTATION CHALLENGES

A properly defined interface between the simulation and optimization tools is only part of the solution, the tools themselves need to conform to the formal domain model and the associated interface. Currently, there remains a need for discrete event simulation language support for specifying control rules and methods in such a way that they can provide a pathway from design and analysis of controllers to prototyping and testing to deployment. Limitations for modeling complex and dynamic control behaviors in discrete event simulation tools are discussed as side-notes in (Jeong 2000, Van der Zee 2006, Yan and Wang 2007)

Adding to these challenges, modeling complex control behavior typically is embedded into the actuator block, e.g. sequencing logic is embedded within the queue block with a pre-defined list of control rules. In the case of sequencing control behavior and the queue 'actuator', modeling more complex control methods requires a more transparent queue block or component, a more general method to specify the sequence of entities in the queue, e.g. using an index variable rather than a sort key, and the ability to change the priority rule throughout the simulation run.

These challenges are illustrated in Figures 5 and 6. While the FSM allows for a consistent control behavior representation, the actuator mechanisms (the queue itself) require different configurations depending on the behavior it is expected to execute. To dynamically change the priority rule during the simulation run, Figure 5 illustrates an execution mechanism that uses one queue for each priority rule and allows the queued tokens to flow into the queue that implements the currently selected priority. To increase the transparency of the queued tokens, Figure 6 divides the tokens into several classes, each with its own individual queue. Therefore the control behavior can use each token's customer class, job type, and current age in the system to decide which token to service next. This challenge could be addressed by increasing the transparency of queues and allowing the control behavior to inspect each token in the queue, or at least the data associated with each token.

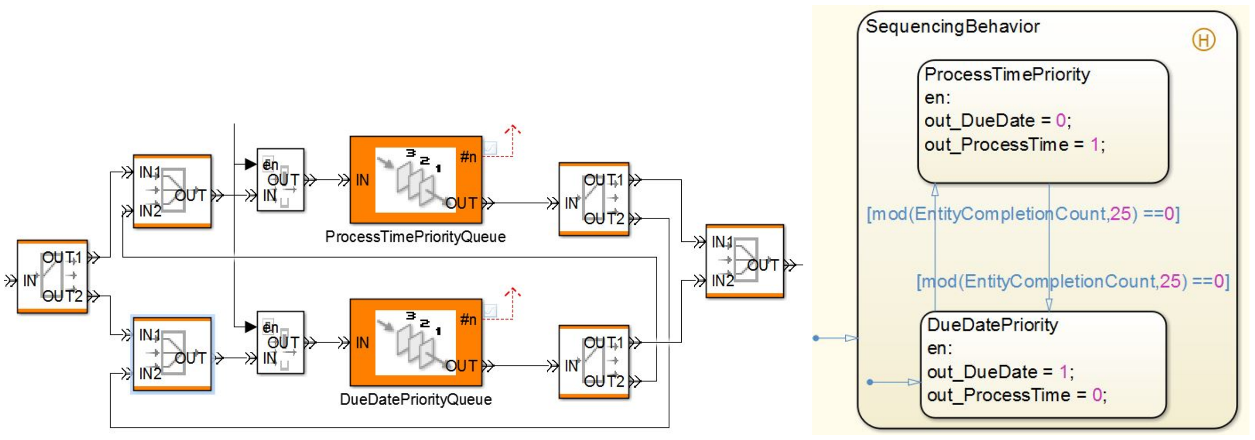


Figure 5: Switching dynamically between priority rules using a state machine to enable a priority queue.

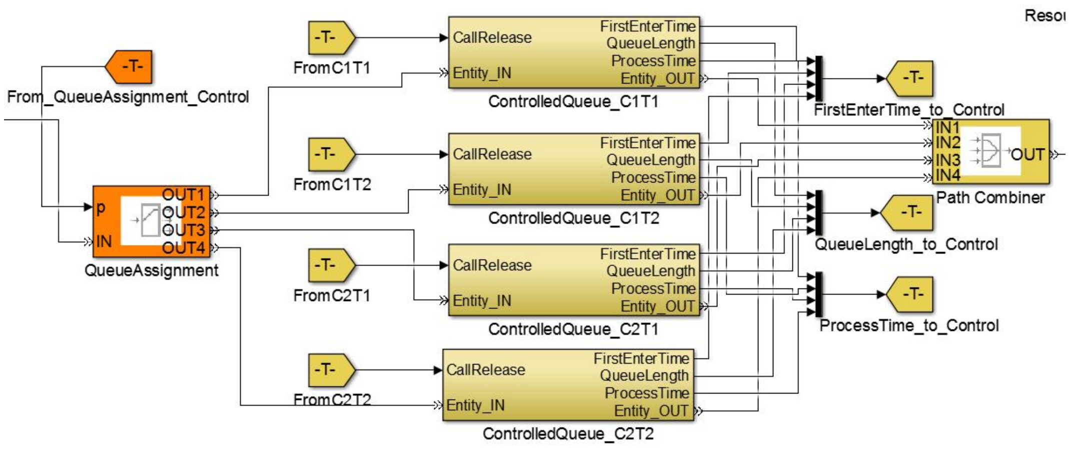


Figure 6: Implementing the complex dynamic sequencing behavior requires unique, and ad-hoc, queuing mechanisms.

The obvious solution to many of the limitations of existing COTS tools is to develop a custom simulation tool for a particular application. However, this approach has the same fundamental shortcoming as discussed above, albeit for a different reason. Without a formal model of the system behavior and control, the simulation model is built with implicit assumptions about the control behavior and the implementation and execution mechanisms are written in an ad-hoc manner. This approach limits the reusability of the resulting analysis model and tool. Moreover, the specificity of any particular solution may not conform to a broader architecture for the domain.

5 CONCLUSION

To simultaneously address the complexity and scale of DELS, simulation and optimization offer complementary views of the system model; simulation is more adept at capturing and evaluating the dynamic behavior of the system, while optimization is more effective at tackling the scale and searching the design space of the system. In order to maximize the effectiveness of these two methods, we have proposed a modeling methodology that can improve interoperability between these two analysis methods and their associated tools. This modeling methodology relies on facilitating semantic interoperability by using a formal model of the domain to define the interface between the simulation and optimization tools. However, there remains a modeling gap in the context of operational control for DELS. To address this gap, in this paper we have proposed a model of operational control that can bridge from the decision variable in the optimization model to the control behavior and execution mechanism required by the simulation model. As noted in section 4, there is ongoing work on identifying implementation gaps and developing corresponding analysis methods and tools that conform to this model of operational control.

ACKNOWLEDGMENTS

This research effort has been sponsored by the National Institute of Standards and Technology under grant No. 70NANB15H234.

REFERENCES

Ahmed, M. A., and T. M. Alkhamis. 2009. “Simulation optimization for an emergency department healthcare unit in Kuwait”. *European Journal of Operational Research* 198 (3): 936–942.

Alabas, C., F. Altiparmak, and B. Dengiz. 2002. “A comparison of the performance of artificial intelligence techniques for optimizing the number of kanbans”. *Journal of the Operational Research Society*:907–914.

Andrews, R., J. Diederich, and A. B. Tickle. 1995. “Survey and critique of techniques for extracting rules from trained artificial neural networks”. *Knowledge-based Systems* 8 (6): 373–389.

Azadivar, F., and G. Tompkins. 1999. “Simulation optimization with qualitative variables and structural model changes: A genetic algorithm approach”. *European Journal of Operational Research* 113 (1): 169–182.

Branke, J., S. Nguyen, C. W. Pickardt, and M. Zhang. 2016. “Automated design of production scheduling heuristics: a review”. *IEEE Transactions on Evolutionary Computation* 20 (1): 110–124.

Cheng, L., and M. A. Duran. 2004. “Logistics for world-wide crude oil transportation using discrete event simulation and optimal control”. *Computers & Chemical Engineering* 28 (6): 897–911.

Costa, A., G. Celano, S. Fichera, and E. Trovato. 2010. “A new efficient encoding/decoding procedure for the design of a supply chain network with genetic algorithms”. *Computers & Industrial Engineering* 59 (4): 986–999.

Davis, W., A. Jones, and A. Saleh. 1992. “Generic architecture for intelligent control systems”. *Computer Integrated Manufacturing Systems* 5 (2): 105–113.

Ding, H., L. Benyoucef, and X. Xie. 2009. “Stochastic multi-objective production-distribution network design using simulation-based optimization”. *International Journal of Production Research* 47 (2): 479–505.

Fu, M. C., G. Bayraksan, S. G. Henderson, B. L. Nelson, W. B. Powell, I. O. Ryzhov, and B. Thengvall. 2014. “Simulation optimization: A panel on the state of the art in research and practice”. In *Proceedings of the 2014 Winter Simulation Conference*, edited by A. Tolk, S. D. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, 3696–3706. Piscataway, NJ, USA: IEEE Press.

Gamma, E., R. Helm, R. Johnson, and J. Vlissides. 1994. *Design patterns: elements of reusable object-oriented software*. Pearson Education.

Grega, W. 1999. “Hardware-in-the-loop simulation and its application in control education”. In *Frontiers in Education Conference, 1999. (FIE’99).*, Volume 2, 1286–7. IEEE.

Holloway, L. E., B. H. Krogh, and A. Giua. 1997. “A survey of Petri net methods for controlled discrete event systems”. *Discrete Event Dynamic Systems* 7 (2): 151–190.

Huyet, A., and J. Paris. 2004. “Synergy between evolutionary optimization and induction graphs learning for simulated manufacturing systems”. *International Journal of Production Research* 42 (20): 4295–4313.

Iassinovski, S., A. Artiba, and C. Fagnart. 2008. “A generic production rules-based system for on-line simulation, decision making and discrete process control”. *International Journal of Production Economics* 112 (1): 62–76.

Jeong, K.-Y. 2000. “Conceptual frame for development of optimized simulation-based scheduling systems”. *Expert Systems with Applications* 18 (4): 299–306.

Kapanoglu, M., and M. Alikalfa. 2011. “Learning IF–THEN priority rules for dynamic job shops using genetic algorithms”. *Robotics and Computer-Integrated Manufacturing* 27 (1): 47–55.

Kapuscinski, R., and S. Tayur. 1999. “Optimal policies and simulation-based optimization for capacitated production inventory systems”. In *Quantitative Models for Supply Chain Management*, 7–40. Springer.

Kim, M. H., and Y.-D. Kim. 1994. “Simulation-based real-time scheduling in a flexible manufacturing system”. *Journal of Manufacturing Systems* 13 (2): 85–93.

Kusiak, A., and M. Chen. 1988. “Expert systems for planning and scheduling manufacturing systems”. *European Journal of Operational Research* 34 (2): 113–130.

Lee, C.-Y., S. Piramuthu, and Y.-K. Tsai. 1997. “Job shop scheduling with a genetic algorithm and machine learning”. *International Journal of Production Research* 35 (4): 1171–1191.

Leong, S., Y. T. Lee, and F. Riddick. 2006. “A core manufacturing simulation data information model for manufacturing applications”. In *Simulation Interoperability Workshop, Simulation Interoperability and Standards Organization*, 1–7.

Mathworks 2015. *SimEvents 2015b User’s Manual*. 4.4.11 ed. Mathworks.

Mönch, L., P. Lendermann, L. F. McGinnis, and A. Schirrmann. 2011. “A survey of challenges in modelling and decision-making for discrete event logistics systems”. *Computers in Industry* 62 (6): 557–567.

Mouelhi-Chibani, W., and H. Pierreval. 2010. “Training a neural network to select dispatching rules in real time”. *Computers & Industrial Engineering* 58 (2): 249–256.

Nie, L., X. Shao, L. Gao, and W. Li. 2010. “Evolving scheduling rules with gene expression programming for dynamic single-machine scheduling problems”. *The International Journal of Advanced Manufacturing Technology* 50 (5-8): 729–747.

Pasupathy, R., and S. G. Henderson. 2011. “SimOpt: A library of simulation optimization problems”. In *Proceedings of the 2011 Winter Simulation Conference*, edited by S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, 4075–4085. Piscataway, NJ, USA: IEEE.

Pierreval, H., and N. Mebarki. 1997. “Dynamic scheduling selection of dispatching rules for manufacturing system”. *International Journal of Production Research* 35 (6): 1575–1591.

Piltan, F., S. Emamzadeh, Z. Hivand, F. Shahriyari, and M. Mirazaei. 2012. “PUMA-560 robot manipulator position sliding mode control methods using MATLAB/SIMULINK and their integration into graduate/undergraduate nonlinear control, robotics and MATLAB courses”. *International Journal of Robotics and Automation* 3 (3): 106–150.

Rosenblatt, M. J., Y. Roll, and D. Vered Zyser. 1993. “A combined optimization and simulation approach for designing automated storage/retrieval systems”. *IIE Transactions* 25 (1): 40–50.

Scholten, B. 2007. *The road to integration: A guide to applying the ISA-95 standard in manufacturing*. ISA.

SCOR 2012. “SCOR: The Supply Chain Reference Model Version 11.0”.

Shirazi, B., I. Mahdavi, M. Solimanpur et al. 2010. “Development of a simulation-based intelligent decision support system for the adaptive real-time control of flexible manufacturing systems”. *Journal of Software Engineering and Applications* 3 (07): 661.

Sivakumar, A. I. 1999. “Optimization of a cycle time and utilization in semiconductor test manufacturing using simulation based, on-line, near-real-time scheduling system”. In *Proceedings of the 1999 Winter*

Simulation Conference, edited by P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, 727–735. ACM.

Sprock, T. 2015. *A Metamodel of Operational Control of Discrete Event Logistics Systems (DELS)*. Ph. D. thesis, Georgia Institute of Technology, Atlanta, GA.

Sprock, T., and L. F. McGinnis. 2015. “A simulation optimization framework for discrete event logistics systems (DELS)”. In *Proceedings of the 2015 Winter Simulation Conference*, edited by L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rossetti, 2776–2787. Piscataway, NJ, USA: IEEE Press.

Syarif, A., Y. Yun, and M. Gen. 2002. “Study on multi-stage logistic chain network: A spanning tree-based genetic algorithm approach”. *Computers & Industrial Engineering* 43 (1): 299–314.

Truong, T. H., and F. Azadivar. 2005. “Optimal design methodologies for configuration of supply chains”. *International Journal of Production Research* 43 (11): 2217–2236.

Valavanis, K. P. 1990. “On the hierarchical modeling analysis and simulation of flexible manufacturing systems with extended petri nets”. *IEEE Transactions on Systems, Man, and Cybernetics* 20 (1): 94–110.

Van der Zee, D. 2006. “Modeling decision making and control in manufacturing simulation”. *International Journal of Production Economics* 100 (1): 155–167.

Xu, J., B. L. Nelson, and J. Hong. 2010. “Industrial Strength COMPASS: A Comprehensive Algorithm and Software for optimization via Simulation”. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 20 (1): 3.

Yan, Y., and G. Wang. 2007. “A job shop scheduling approach based on simulation optimization”. In *Industrial Engineering and Engineering Management, 2007 IEEE International Conference on*, 1816–1822. IEEE.

Yang, T., Y. Kuo, and I. Chang. 2004. “Tabu-search simulation optimization approach for flow-shop scheduling with multiple processors - a case study”. *International Journal of Production Research* 42 (19): 4015–4030.

Zhang, Q., and H. Li. 2007. “MOEA/D: A multiobjective evolutionary algorithm based on decomposition”. *IEEE Transactions on Evolutionary Computation* 11 (6): 712–731.

Zhang, T., and O. Rose. 2014. “Simulation-based dispatching in job shops”. In *ASIM 2014, 22. Symposium Simulationstechnik 2014 - Berlin*. ASIM.

Zhou, G., H. Min, and M. Gen. 2002. “The balanced allocation of customers to multiple distribution centers in the supply chain network: a genetic algorithm approach”. *Computers & Industrial Engineering* 43 (1): 251–261.

AUTHOR BIOGRAPHIES

TIMOTHY SPROCK is a Postdoctoral Fellow in the Stewart School of Industrial and Systems Engineering at Georgia Tech. He holds a PhD in Industrial Engineering from Georgia Tech. His research interests include model-based systems engineering methodologies for discrete event logistics systems. His email address is tsprock3@gatech.edu.

LEON F. MCGINNIS is Professor Emeritus in the Stewart School of Industrial and Systems Engineering at Georgia Tech and founder of the Keck Virtual Factory Lab. He is internationally known for his leadership in the material handling research community and his research in the area of discrete event logistics systems. A frequent speaker at international conferences, he has received several awards from professional societies for his innovative research, including the David F. Baker Award from IIE, the Reed-Apple Award from the Material Handling Education Foundation, and the Material Handling Innovation Pioneer award from Material Handling Management Magazine. His current research explores the use of formal systems modeling methods to support systems engineering of discrete event logistics systems. Professor McGinnis is a Fellow of the Institute of Industrial Engineering. His e-mail address is leon.mcginnis@gatech.edu.