15th Annual Conference on Systems Engineering Research
Disciplinary Convergence: Implications for Systems Engineering Research

Eds.: Azad M. Madni, Barry Boehm
Daniel A. Erwin, Roger Ghanem; University of Southern California
Marilee J. Wheaton, The Aerospace Corporation
Redondo Beach, CA, March 23-25, 2017

Patterns for modeling operational control of discrete event logistics systems (DELS)

Timothy Sprock[a]
[a] National Institute of Standards and Technology, timothy.sprock@nist.gov

Abstract

Designing smart operational control systems for Discrete Event Logistics Systems (DELS) requires a standard description of control behaviors executed at the operational management level of DELS control. In this paper, we propose a set of patterns for modeling the operational control mechanisms, organized by classes of control questions that all DELS must be able to answer. The pattern for each control question includes an analysis-agnostic functional definition of the control problem for that question, as well as a mapping of the decision variable in that problem to a particular function and execution mechanism in the base system.

Keywords: System Design Methods; Smart Manufacturing; Discrete Event Logistics Systems;

1. Introduction

Discrete Event Logistics Systems (DELS) are a class of dynamic systems that transform discrete flows through a network of interconnected subsystems [1]. These include systems such as supply chains, manufacturing systems, transportation networks, warehouses, and health care delivery systems. Traditionally, each specialized kind of DELS has been regarded as a distinct class of systems requiring its own dedicated research and development. However, these systems share a common abstraction, i.e. *products* flowing through *processes* being executed by *resources* configured in a *facility* (PPRF), and they appear together in integrated models of the enterprise. For example, production systems might integrate storage and fulfillment capabilities as well as material handling and transportation systems, and supply chains might integrate flows between warehouses, transportation systems, and manufacturing or health care facilities.

The increasing size, integration, and complexity of next-generation smart DELS requires more robust engineering design methods. Fundamental to more robust design methodologies are explicit system specifications and more powerful search and decision-support algorithms; see [2] for an example in the warehousing context. Next-generation smart control systems must integrate more information feedback from sensors in the plant and from global information systems, as well as accommodate greater automation. These systems are more software intensive than traditional plant designs, which have focused primarily on hardware selection and configuration. A standard description of operational control would enable development of a uniform interface to decision tools, supported by interoperable, or plug-and-play, analysis tools. However, despite progress on modeling the structure and behavior of DELS, a standard representation of operational control problems for DELS remains a challenge.

This paper presents a set of patterns for modeling the operational control mechanisms for DELS. These patterns include an analysis-agnostic description of each control problem that is used to connect the controller's decision problem to the corresponding actuator function and execution mechanism in the base system. The rest of the paper is organized as follows: Section 2 provides context for modeling operational control in DELS, Section 2.1 provides an informal introduction to the control problems, and Section 2.2 describes a simple form for defining control patterns. Then Section 2.3 uses the pattern form to capture each of the functional control mechanisms and provides a reference architecture for assembling control components. Then Section 2.4 describes a concrete application of the patterns to specifying a smart manufacturing system. Finally, Section 3 discusses future directions.

2. Modeling Operational Control in DELS

Operational control is the manipulation of flows of tasks and resources through a system in real-time, or near real-time. Each task requires or authorizes a DELS to use its resources to complete a portion of the process plan contained in that task. Operational control consists of multiple mechanisms, each including a function in the controller that prescribes actions to be taken and an actuator in the base system (or plant) that executes the prescribed action. A model of operational control is part of a broader approach to modeling DELS that includes a domain specific language, a reusable component library, and a reference architecture [3,4]. This broader model is organized into three layers: the structure layer that captures flow networks, process networks, and relationship networks; the behavior layer that describes each DELS using product, process, resource, and facility (PPRF) concepts; and finally, the control layer, which is the focus of this paper.

A standard analysis-agnostic representation of each control problem is necessary to bridge the gap between the system model and analysis models that support design methods and operational decision-making for the system, including statistical (description), discrete event simulation (prediction), and mathematical programming (optimization) models. This standard representation of operational control is a set of patterns in the DELS reference architecture [5]. These patterns for operational control can describe the control of existing systems, as well as guide design of control for new or modified systems.

2.1. The Operational Control Questions

The operational control layer of the DELS reference architecture is based on a set of control problems organized by questions posed by the base system to the controller and corresponding answers that prescribe control actions for the actuators in the base system to execute [4]. Each control question encapsulates a single function of a controller to manipulate the flows of tasks and resources (note that controllers may leverage several functions jointly, i.e. answer several questions together as is the case for scheduling). These control questions are:

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) “Where (to which DELS for what process) should the task be sent after it completes the required processing at the current DELS?” (*routing*);
5) “When, and to which state, does the state of a resource need to be changed?” (*change in resource capacity or capability*).

Figure 1 illustrates the interaction between the base system and controller (control questions and answers), as well as the control execution mechanisms (actuator components in the base system). For more discussion on the functional requirements of the controller itself (how it answers the control questions), see [4,6].

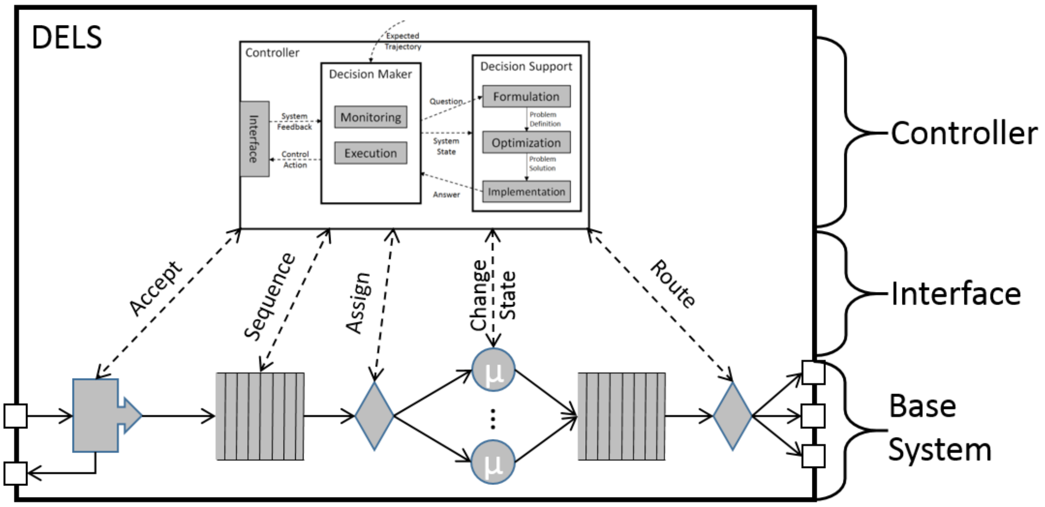

Figure 1. This illustrates the control execution mechanisms (the actuators in the base system), the functional architecture of the controller, and the interaction between the base system and controller (control questions and answers).

For example, production systems use manufacturing resources to service orders or jobs (a kind of task), leading to the following control problems:

1) Admission decides whether or not to accept an order from a customer. The decision may evaluate available production capacity, including raw material inventory on-hand, operator availability, and the current state of resources.
2) Sequencing orders includes decisions such as prioritization (of some customers' orders over others), coordination (of orders outbound to the same customer), batching (similar orders together for efficient processing or transport), delaying (service of an order until a future period or backordering), and splitting (an order into smaller lots to be processed over time).
3) Resource assignment refers to many interrelated problems including assigning scare resources to orders. Manufacturing resources may include labor, critical processing equipment, or material handling equipment. Orders may also require assignment of auxiliary resources such as tools, fixtures, and storage locations to enable process execution.
4) Routing physically or virtually directs orders to resource locations in a facility as required by product's process plan. The routing decision also accommodates unplanned auxiliary processing steps such as exception handling, quality inspection, or unexpected buffer storage, as well as routing optimization for automated guided vehicles (AGVs).
5) Changing the capability or capacity of resources includes replenishment of input material stocks, maintenance on automated systems, changing set-ups or tooling for machines, or anticipatory movement and pre-positioning of inventory or vehicles.

2.2. A form for defining operational control patterns

Operational control patterns bring together several representations of each control problem to describe the problem in a standard way. A simple form is used in this paper to define the patterns, derived from a function-behavior-structure representation [7,8]. The components of the form are:

- **Name**: Colloquial identifier of the control problem being addressed (the literature uses various names for the control problems).
- **Question**: Domain-independent, informal "what should I do?" kind of question that the base system poses to the controller (the answer to which is an appropriate control action).
- **Control Function**: Transformation, or mapping, of system objects and their state data to actionable control decisions. This transformation formalizes the control question and answer, where the question identifies an applicable control function, and the answers are the control decisions to be executed by the base system. The transformation specifies the functional interface, or signature, of conforming analysis models that answer this particular control question.
- **Decision Expression**: A formulation of the control function in terms of one or more binary decision variables (0/1) representing the decision, for use in an optimization analysis model. This part of the pattern describes the control action to be taken when the decision variable has 1 as a value.
- **Actuator Function**: Expected effect of the actuator in the base system, for use in simulation models. This is how the control function is carried out by the actuator.
- **Actuator**: Abstract actuator that is capable of carrying out the actuator function. The actuators in this paper are selected from common discrete event simulation modeling components.

The actuator function and actuator itself are used to specify the base system model and corresponding simulation models. The decision variable defines the intent of the optimization model (what question can it answer) that is used to provide decision support. Typically, the actuator and decision variable are developed independently without a shared representation of the control mechanism they are both expressing. The control function provides a common abstraction for these elements to follow, improving interoperability between simulation and optimization analysis models and between the decision support system and the base system it is guiding.

2.3. Patterns for modeling operational control of DELS

In Figure 2, the form described in section 2.2 is used to specify how to answer each operational control question from section 2.1:

1) Admission determines which tasks the controller should admit into the base system. The corresponding decision variable is a yes/no admission choice for each task. This decision is implemented by a function that adds an accepted task to the system's queue. This function is executed by a gate actuator, which is opened, or closed, according to the variable value.
2) Sequencing determines the order, or partial order, that admitted tasks will be serviced by the system. The corresponding decision variables determine the index (position) of each task in the queue. The ordering is implemented by a function that sorts the tasks by an index determined by the decision problem and is executed by the queue where tasks are waiting for service. The task at the head of queue is serviced next.
3) Assignment matches tasks to resources, or partitions the tasks into resource-specific subsets, based on resource capabilities. The decision variable matches tasks to resources, which is implemented by a function that places the task, either virtually or physically, into the assigned resource's queue. Depending on the modeling paradigm, the assignment can be executed by a switch that directs the flow of the task to the resource, which is common in resource-oriented

modeling, or it can be executed by seizing the resource from a pool.

4) To route a task for completion of its process plan, the current DELS (or the task itself) must determine where to send the task after the current DELS has completed the requested processes. Specifically, the routing decision identifies the next process required by the task's process plan (*nextProcess*) and selects a suitable DELS to perform that process (*targetDELS*). The actuator function is the composition of two functions: f is responsible for evaluating the task's process plan to determine the next required process (*nextProcess*), while g is responsible for finding a DELS (*targetDELS*) that is capable of executing the next required process for the task, via, e.g., by directory lookup or call for proposal. These two functions are not necessarily executed in any particular order; i.e. DELS can be solicited to perform each potential process before resolving alternative paths in the process plan or resolve the alternatives then find suitable DELS. This

	Admission	**Sequencing**
Question	"Should the task be served?"	"When (in what order) should the tasks be served?"
Control Function	$Admit\text{:Task} \rightarrow \mathbb{B}$	$Sequence\text{:Task} \rightarrow \mathbb{N}$
Decision Expression	$x_i = 1,$ if task i is admitted to the system	$x_{ik} = 1,$ if task i is serviced k^{th}
Actuator Function	$\text{Admit(Task)} := \{\text{System. TaskSet}\} \cup \text{Task}$	$\text{Sequence(TaskSet)} := \text{sort(TaskSet, Index)} = \text{TaskSet}'$
Actuator	Abstract Gate	Abstract Queue

	Assignment	**Routing**
Question	"Which resource should serve the task?"	"Where (to which DELS for what process) should the task be sent after this DELS?"
Control Function	$Assign\text{: TaskSet} \times \text{ResourceSet} \rightarrow \mathbb{B}^{\lvert T \rvert \times \lvert R \rvert}$	$Route\text{: Task.ProcessPlan} \rightarrow \text{Process} \times \text{DELS}$
Decision Expression	$x_{im} = 1,$ if task i is assigned to resource m	$x_{OD} = 1,$ if the task is output to DELS D for process O
Actuator Function	$\text{Assign(Task, Resource)} := \{\text{Resource. TaskSet}\} \cup \text{Task}$	$\text{nextProcess} = f(\text{Task.processPlan})$ $\text{targetDELS} = g(\text{nextProcess})$ $Route(\text{Task}) = g \circ f \vee f \circ g$
Actuator	Abstract Switch or Resource Seize	Abstract Switch

	Change Capability State	**Change Capacity State**
Question	"Should the capability state of a resource be changed?"	"Should the capacity state of a resource be changed?"
Control Function	$ChangeState\text{: Resource.State} \rightarrow \text{newState}$	$ChangeState\text{: Resource.State} \rightarrow \text{newState}$
Decision Expression	$\mathrm{X}_{\mathrm{ij}}^{\mathrm{m}} = 1,$ if resource m is changed from state i to state j	$\mathrm{X}_{\mathrm{ij}}^{\mathrm{m}} = 1,$ if resource m is changed from state i to state j
Actuator Function	$changeService(\text{Process}) := \text{Resource.CurrentService} \rightarrow \text{Process}$ $changeLocation(\text{Location}) := \text{Resource.CurrentLocation} \rightarrow \text{Location}$	$increaseCapacity(\text{Quantity}) := \text{Resource.CapacityMeasure} + \text{Quantity}$ $decreaseCapacity(\text{Quantity}) := \text{Resource.CapacityMeasure} - \text{Quantity}$
Actuator	Abstract `Non-Preemptive Resource Setup' Task	Abstract `Change Resource Pool Size' Task

Figure 2. The patterns for operational control specification in DELS

function is executed by an abstract switch that outputs the task to a particular flow interface, which is connected to the selected target DELS.

5) Deciding to change the capability or capacity of a particular resource uses an abstraction of state to unify the models for answering this control question. The decision variable determines whether to transition resource *m* from state *i* to state *j,* where state is an abstraction for capability or capacity. The pattern for capability states describes the function, process, or service that a discrete state resource can execute at a particular time or a geographic location, i.e. serving tasks at a particular location. The pattern of capacity states describes the amount of work that can be assigned to a particular resource. The functions that change capability and capacity are abstractly modeled as set-up (*changeService*) and replenishment behaviors (*increaseCapacity*), respectively. Both behaviors are executed by generating an overhead task, which is accepted, scheduled, and executed by some other resource, such as an operator, maintenance resource, or procurement system.

The abstract actuators corresponding to control questions in Figure 2 are model library components for constructing system models. Figure 3 uses these components to formalize the process illustrated in Figure 1. Each component enables the DELS to control the flow of tasks and resources through the system. In Figure 3, a new task enters through the *inTask* port of the DELS and is handled by the *admissionGateway,* a gate type resource that decides whether to admit the task or not. The *admitTask* port (small rectangle on the border of the admissionGateway) interfaces with the controller, which provides the gate with a yes or no (Boolean) decision for each task. The admitted task then flows (filled triangle) into the queue containing the system's *taskSet*. The *sequenceIndex* port interfaces with the controller, which provides the queue with a sequence for the tasks stored in the queue. In this model, resources may be seized from a local *resourceSet* that is owned by the DELS, or may be requested and seized from outside the system through the *ioDELSResource* interface. The *resourceAssignment* port interfaces with the controller, which provides the set of resources that are assigned to serve the task. Once necessary resources have been acquired, the task and resource flow through the remaining internal control processes and actuators as follows:

- The task and resources flow to the *Process* node.
- After processing is complete, the reusable resources are released (*releaseResources)* back to the system's central *resourceSet* or out of the system through the *ioResource* port (flow not depicted).
- The task then enters an outbound queue (*completedTaskSet*) that stores completed tasks waiting to form move batches or the next DELS to be available.
- Finally, the task departs the completed task queue, is routed by the switch (*routing*) to its next DELS, and departs through the appropriate *outTask* port. The *nextNode* port interfaces with controller, which provides the target DELS (which output port) for the task to be routed.

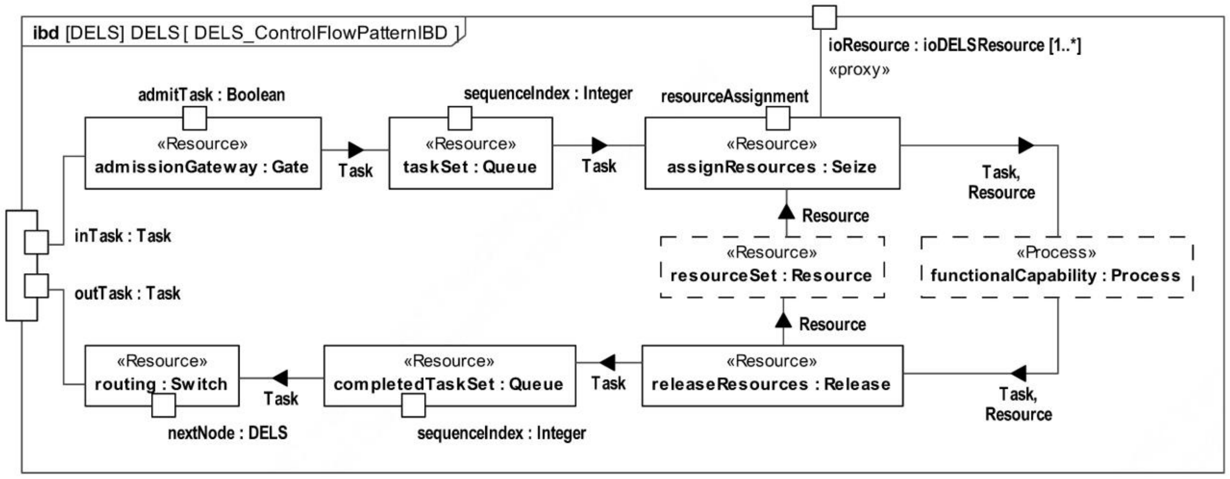

Figure 3. This shows how abstract actuators can be assembled to control the flow of a task through the system.

2.4. Applying the Patterns – Towards a Smart Manufacturing System Use Case

Designing operational control for DELS includes configuring operational control logic and selecting concrete actuators in the base system that execute the prescribed control actions. The patterns in section 2.3 describe operational control problems independently of DELS domains, enabling them to apply to all domains and improve interoperability of base system models with analysis models supporting control decisions, such as optimization. Analysis models are often constructed at a high-level of abstraction, but designing a system to execute control decisions requires selecting specific equipment to carry out the function of the abstract actuator (embodiment design). Some examples of concrete actuators for a smart manufacturing system are:

1) The admission gate might be a robotic arm that retrieves the physical workpiece, work in process, and other input resources associated with a task, from an AGV or pneumatic pusher that moves tasks from a centralized conveyor onto the system's local conveyor.
2) Sequencing, and its associated abstract queue for storing tasks waiting for service, might be done by a range of technologies with varying capabilities for executing complex control behaviors. For example, some non-automated storage solutions might only be capable of simple control behaviors; for example, a gravity-fed conveyor might only be capable of enforcing a First In First Out (FIFO) discipline. Some technologies may not be capable of enforcing any sequencing discipline at all; for example, a simple storage rack requires the operator, possibly with the aid of pick lights, to execute the desired sequencing discipline. Simple storage technologies might be augmented with an automated technology, such as a robotic arm capable of picking items from slots, to create a combined system that operates like an automated storage and retrieval systems (ASRS).
3) Assignment of concrete resources to each task depends on whether the task is being brought to the resource or if the resource is being seized and brought to the task. Many systems use both mechanisms. In the case of stationary equipment, such as in a work cell, the assignment mechanism might direct a task into the equipment's queue via pneumatic switch on a conveyor. For resources in a central pool of available discrete units, e.g., input materials, fixtures, or tools, the assignment actuator might be implemented as a robotic arm or AGV that removes the resource from a central buffer and transports it to the work station.

4) Routing tasks from the system often complements admission control and might rely on technologies similar to those that bring tasks into the system. However, in material handling systems where an AGV (or non-automated worker) deliver the task, the routing behavior must first summon an AGV to the system. Then a robotic arm, or similar mechanism to the admission actuator, can place the task onto the AGV.
5) Change state decisions generate an overhead task for the system, to perform set-up or maintenance, reposition a tool or vehicle, order additional inventory, etc. These overhead tasks are assigned, scheduled, and executed by their respective systems, e.g. maintenance, material handling, or procurement, which follow the same control pattern described in section 2.3.

3. Conclusions and Future Work

To facilitate design and analysis of dynamic, intelligent operational control methods for next-generation DELS, this research identifies a set of control problems, posed as questions to a controller, and patterns for defining the functional, behavioral, and structural aspects of these problems. The patterns use functional descriptions of operational control to connect the base system (plant) model with decision support models, such as optimization and simulation.

The standard description of operational control that is captured by the patterns in this paper supports the development of interoperable, or plug-and-play, analysis tools to answer the control questions. This unifying abstraction of operational control also enables a uniform architecture for each kind of controller, which is especially important for a flexible control architecture and transitioning from traditional centralized, hierarchical control to adaptive, decentralized or holonic architectures [9]. This vision contrasts with each DELS having a different controller architecture depending on its responsibilities, requirements, and the broader control hierarchy selected.

A future goal is formalizing the definitions of operational control for DELS that are documented here informally using patterns. The objective is to formalize the control questions into a canonical set that rigorously partitions the set of all DELS control problems into equivalence classes with a formal equivalence relation (~) based on the functional mapping and associated interface definition. Additional formalization is also required for the mapping between the functional definition and the analysis components.

Future extensions to the control patterns are expected to include concrete system modeling objects as a model library of components, an interface definition for optimization methods, and representative simulation modeling components. Additional reference implementation patterns will be developed that describe integration of simulation components with operational control methods (optimization). The goal is to verify and validate control in simulation and port the logic to a real system, as is common in other engineered systems.

Acknowledgements

The author thanks Conrad Bock for his helpful conversations and comments.

References

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

2. McGinnis, L. and Sprock, T., 2016. Toward an engineering discipline of warehouse design. *14th International Material Handling Research Colloquium.*
3. Thiers, G., 2014. A model-based systems engineering methodology to make engineering analysis of discrete-event logistics systems more cost-accessible. Ph.D thesis, Georgia Institute of Technology, Atlanta, GA. https://smartech.gatech.edu/handle/1853/52259
4. Sprock, T., 2016. A metamodel of operational control for discrete event logistics systems. Georgia Institute of Technology. Ph.D thesis, Georgia Institute of Technology, Atlanta, GA. https://smartech.gatech.edu/handle/1853/54946
5. Cloutier, R.J. and Verma, D., 2007. Applying the concept of patterns to systems architecture. *Systems engineering*, 10(2), pp.138-154.
6. Sprock, T. and McGinnis, L.F., 2015. A Conceptual Model for Operational Control in Smart Manufacturing Systems. *IFAC-PapersOnLine*, 48(3), pp.1865-1869.
7. Gero, J.S., 1990. Design prototypes: a knowledge representation schema for design. *AI magazine*, 11(4), p.26.
8. Umeda, Y., Takeda, H., Tomiyama, T. and Yoshikawa, H., 1990. Function, behaviour, and structure. *Applications of artificial intelligence in engineering V*,*1*, pp.177-194.
9. Dilts, D.M., Boyd, N.P. and Whorms, H.H., 1991. The evolution of control architectures for automated manufacturing systems. *Journal of manufacturing systems*, 10(1), pp.79-93.