Proceedings of the 2024 Winter Simulation Conference
H. Lam, E. Azar, D. Batur, S. Gao, W. Xie, S. R. Hunter, and M. D. Rossetti, eds.

MODELING OPERATIONAL CONTROL IN DISCRETE-EVENT LOGISTICS SYSTEMS AND THEIR DIGITAL TWINS

Lorenzo Ragazzini[1], Leon F McGinnis[2], Elisa Negri[1], and Marco Macchi[1]

[1]Department of Management, Economics and Industrial Engineering, Politecnico di Milano, Milano, MI, ITALY
[2]Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, USA

ABSTRACT

In manufacturing systems, operational control plays a crucial role in ensuring proper functioning and efficiency. Traditional approaches to modeling operational controllers in simulation often suffer both from inability to properly represent decisions as they are made in real systems and from rigidity and lack of adaptability to changing system requirements and configuration during their design. This paper explores a novel approach to designing and implementing operational controllers in discrete-event simulation based on the introduction of control policies as standardized structures that explicitly integrate operational control decision-making into DES models. Standard structures allow for easy integration with optimization tools, facilitating the exploration of different control policies during the design of a new system. The use of control policies also has important implications for the development of digital twins of manufacturing systems, as it helps align the way systems control is designed with the control of real systems.

1 Introduction

Advances in data analytics and machine learning have driven the scale and scope of Cyber-Physical Systems (CPS). In particular, for the Discrete Event Logistics Systems or DELS (e.g., factories, warehouses, supply chains and similar) the opportunities for developing CPS applications to improve efficiency, reduce costs and mitigate risks are especially appealing. One illustration of this appeal is the rapidly growing interest in "digital twins", or computational models of DELS whose computed results mimic the operational results of their "real twins" throughout their lifecycles.

In the DELS domain, a popular approach to developing a Digital Twin (DT) is discrete event simulation as other time-oriented modeling methods generally can't cope with the scale, number of entities and intricate entity interactions of specific DELS. In developing the simulation elements of a DELS DT, it is relatively straightforward to accurately model the item flows of materials, people, vehicles, etc, in part because those flows are easily observable.

To be truly effective, a DELS DT also must accurately reflect the operational decision-making of its corresponding Real Twin (RT). The traditional queue-server paradigm of early simulation tools is simply not adequate for this purpose. Contemporary simulation tools may provide the capability for creating accurate operational control models, but a generic modeling approach that can guide the representation of such decisions throughout system design and operation is still missing. The stronger the mapping between operational controls implemented in the DT and RT, the more effective the DT will be as a platform for designing DELS while studying their control strategies.

This paper explores an implementation strategy for operational control in a DELS DT, anticipating two use cases. The first is ***system design***, where the DT is an essential tool for evaluating system design alternatives in terms of both base system resources and operational controls. This use case is best supported by an object-oriented approach in which changes to either the base system or the operational controller can be easily implemented in the simulation DT. The second use case is ***system operation***, where the DT is

used to explore the RT future trajectory experimentally under alternative operational control decisions in order to provide the RT with changes to operational control decision processes. In both use cases, the capability for experimentation with different control policies is essential. The proposed approach yields operational controllers that conform to the ISA-95 standard (InTech 2021) and thus arguably are implementable in the corresponding RT.

Section 2 summarizes the operational controller research on which this paper is based. Section 3 describes the use case, the design of a highly automated logistics hub, and section 4 presents the design of the hub's operational controllers in more detail. Section 5 describes the operational controller implementations in AnyLogic™, and how the implementation supports experimentation with alternative operational control policies and parameters. Section 6 is a brief discussion of future research opportunities.

2 Prior research

As described in McGinnis (2019), Ehmke et al. (2011), McGinnis, Huang, and Wu (2006), and Thiers and McGinnis (2011), any system in which discrete units of flow move between resources and are transformed by processes executed by those resources is in the DELS domain. Other works also deal with the intersection of DELS and reliability engineering Cui, Shi, and Wang (2015). An initial effort to develop a Domain-Specific Language (DSL) for describing instances of DELS is described in Sprock et al. (2019). The conceptual and functional design of an operational controller for DELS is explored in Sprock (2016) and Sprock and McGinnis (2015), and in McGinnis, Buckley, and Ali (2021) an implementation is described in the context of a hypothetical parcel logistics hub.

A cornerstone for the work described here is the functional architecture of an operational controller as shown in Figure 1 below. As the operational controller receives state change information from its base system, the *EventDirector* determines what kind(s) of decisions are appropriate. The processes of formulating an analysis, solving, and recommending a task may lead to activating a capability of a base system resource. The work described below pursues the idea that it is equally possible that a parallel decision-making process may be launched that would modify the controller's operational logic.

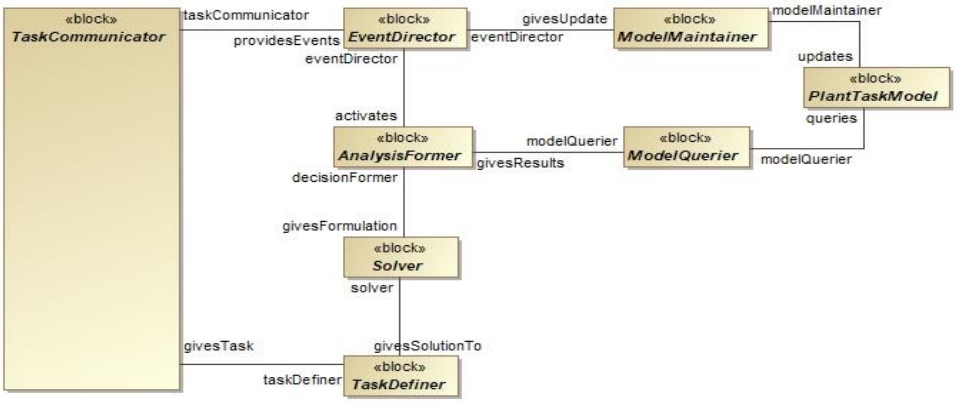


Figure 1. Controller Functional Architecture, from McGinnis (2019).

Note that Figure 1 is a functional architecture and not a process model. The kinds of analyses implemented could include assessments of operational effectiveness and the kinds of tasks defined could include changes to the operational controller itself, such as changing policies or parameters within policies.

The functional architecture of Figure 1 can be implemented in many different ways. Essential to any implementation are: the controlled system model (either implicit or explicit); maintaining the state of the system model relative to the base system; event-driven changes to the controlled system model and event-

driven decision-making; making operational control decisions based on current system state; and translating those decisions into commands executable by the base system resources.

3 DEMONSTRATION USE CASE

This case was developed from a related research project in Georgia Tech's Physical Internet Center and described in greater detail in Babalou et al. (2021) and Montreuil et al. (2021). This project studied the potential for a completely automated parcel logistics hub utilizing totes to aggregate parcels with the same (perhaps next) destination (hub). In McGinnis, Buckley, and Barenji (2021), a full-scale (over 70,000 totes/day throughput) AnyLogic™ simulation is described which provides both real-time and summary operational performance statistics in an easy-to-consume dashboard.

At an originating hub, parcels are sorted and consolidated into totes by destination hub. Totes are placed into racks which may contain totes for multiple destination hubs. Racks are loaded onto trucks and transported between hubs. Each hub that receives racks will then either cross-dock the rack, if the rack is full and all its totes have the same destination, or will swap totes with other racks to improve the rack's "destination consolidation", i.e., more totes in the rack going to the same destination. When a rack arrives at its destination hub, totes are opened and parcels delivered to their recipients.

The technology for accomplishing destination consolidation employs several kinds of robots. Racks are loaded to and unloaded from trucks by loadbots that travel between the trucks and dock-associated staging cells. Consolidation takes place in shuffle cells which have capacity for eight racks and move totes between racks using a shufflebot. There are buffer cells for temporarily storing racks that have to be moved out of staging cells but have not yet been assigned to a shuffle cell, or for racks that have been moved out of one shuffle cell but not yet assigned to another shuffle cell or to an outbound staging cell. Racks are moved in and out of the buffer and shuffle cells by movebots. In addition, individual totes may be routed between shuffle cells using smaller totebots.

Over the course of a day, throughput in the hub will vary as there are periods when few if any trucks will be arriving or departing, and periods when many trucks will be arriving and after some processing time lag, many trucks will be departing. Operationally, this presents significant challenges for managing resources to maintain adequate levels of service, i.e., timely transfer from input dock to output dock.

Operational control in this parcel logistics hub simulation is designed based on the principles outlined in McGinnis (2019), Sprock (2016), and Sprock and McGinnis (2015). Relevant for the present discussion are the operational control decisions associated with opening and closing shuffle cells, and with the planning and execution of shuffle moves within a given shuffle cell. An operational controller for a shuffle center (containing many shuffle cells) determines when additional shuffle cells should be put in operation, or when some shuffle cell can be taken off-line, based on current hub throughput requirements. Each shuffle cell has its own operational controller to manage the movement of totes by the shufflebot, and it attempts make the movement of the shufflebot efficient, according to some appropriate metric. The functional architecture of individual controllers follows the suggestion in McGinnis (2019), where there is a function that evaluates state changes in its base system (for the shuffle center the base system is the set of operating shuffle cells; for a shuffle cell the base system is the set of racks, their tote content, the shuffle bot and the state of the interface for totebots) and determines what kind of control decision is appropriate.

In the interest of brevity, this discussion will not address the issue of managing the flow of racks through the hub. For example, when a shuffle cell has capacity to add a rack, it requests a rack to be delivered and a flow control decision is required to determine the origin of said rack. Similarly, when a rack cannot be further consolidated in the shuffle cell as currently configured, it requests the rack be removed, and again, a decision is required as to the destination of said rack. The controller of rack flow follows the same principles as the controllers for shuffle cell opening/closing and shuffle cell operations. Similarly, there is a controller for assigning each rack and tote move to specific movebots and totebots which also follows those same principles but is not discussed in detail here.

4 MODELING SMART CONTROL IN THE LOGISTICS HUB

The goal is to develop a DT as a simulation model capable of supporting decision-making across different system lifecycle phases, particularly design and operation. For this purpose, the DT must be easily adaptable to different design decisions, e.g., different resource decisions or different operational control policies. Because the DELS DSL is object oriented, using it to structure the simulation model behind a DT facilitates the adaptability required for concurrent design of the system in terms of resources and of its control strategies.

In the different lifecycle phases, different reasons for designing an operational controller within a DT emerge, which include:

1. In the design phase, the controller should support accurate modeling of the system behavior. By exploiting the concurrent design of the system and of its controllers, it is possible to ensure that the complexities in the management of the system operation are evident since the early design stages.
2. In the operation phase, the adoption of an operational controller within the DT allows to study the adaptation of the existing control strategies to maintain high system performances under changing conditions, in order to respond to particular controllers that can "self-optimize on the fly" by tuning operational control parameters or by changing operational control policy.

Elaborating on previous literature on operational control, the structure of an operational controller is revised by introducing the elements that allow to build and operate its functional blocks, as depicted in Figure 2.

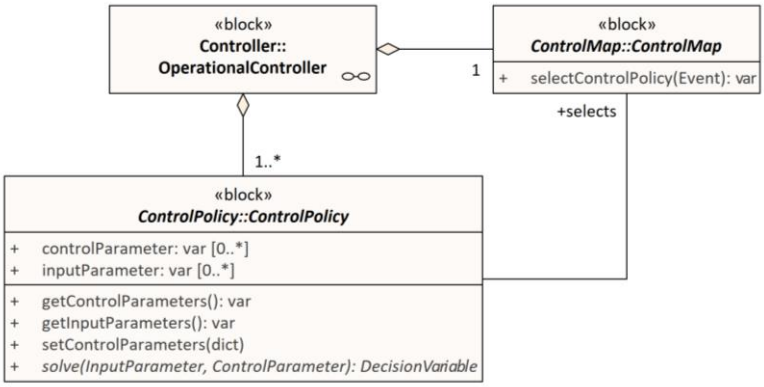


Figure 2: Control policy and control map for operational control.

The three blocks in Figure 2 enable an implementation of *EventDirector*, *AnalysisFormer*, and *Solver* in Figure 1. The *Controller* in Figure 2 includes a *ControlMap* which maps the events to the appropriate *ControlPolicy*. According to the kind of event occurring, and to the relevant data available (based on state data from the *PlantTaskModel*), to the controller, the *ControlMap* selects the *ControlPolicy* to respond to the event. There may be several control policies to select from, and each one may require different input parameters whose values may be determined considering base system state and state trajectory.

The controller executes the selected *ControlPolicy*, also populating the input parameters as required. As a simple example, consider managing a queue between two resources in series. Alternate policies for sequencing work at the second resource might be first-come-first-served, shortest processing time, least remaining slack, highest-value-first, or some weighted combination of criteria. The *ControlMap* function might consider the state of the queue or the utilization of the resource to recommend a policy. The *ControlPolicy* function for some policies may have no tunable parameters, but for the weighted combination of criteria, the value of the weights may be computed based on data collected by the *PlantTaskModel*.

The implementation structure from Figure 2 is the basis for important standardization, specifically of the interfaces to the control map and control policy functions. In the system design scenario, when new control policies are proposed, they can be added to the simulation easily by adding a new control policy function conforming to the interface standard. The control map function will have a new policy as a policy option and the controller will have a new call to the new control policy function.
This approach to implementation also allows structuring the set of control policies as illustrated in Figure 2, where the class of dispatching rule control policies can be elaborated in multiple ways. The *DispatchingRule* object extends the *ControlPolicy* by introducing the sorting method, which is used to order tasks according to some priority value. The specific dispatching rule defines how the priority of each task is computed by specifying the criteria utilized for calculating such priority. For the purpose of this work, FIFO and LIFO dispatching rules are shown as implementations of *DispatchingRule* block. Other more sophisticated dispatching rules could be implemented in a similar way.

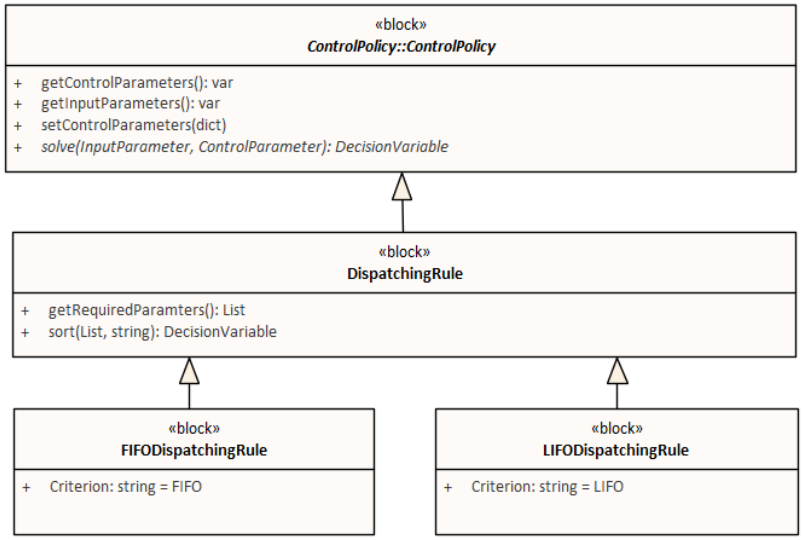


Figure 3: Dispatching rule control policies.

4.1 Control Policies

This section is devoted to specifying how control policies fit into the operational controller functional architecture, allowing it to work.
The control policy is designed to accommodate the key elements required by the controller for its decision-making activity and it is constituted of three main elements:

1. **Control function**. It is the main element of the control policy and it includes a set of instructions, routines, or procedures that can be executed by the controller. This function is the core of the control policy itself as it allows mapping the input parameters to a set of decision variables. The control function can be customized to change the behavior of the control policy and thus the behavior of the whole system, without requiring redesign of the controller itself.
2. **Input parameters**. Input parameters provide information on the tasks and the state of the resource required to make control decisions.
3. **Control parameters**. Control parameters include all the data which can be customized and used by the controller to modify the control function. Examples of this include the number of kanbans for card-based production control or the objective value of an optimization function. These parameters can be used to ease the customization of control policies providing a simple way to change the policy overall behavior.

The control policy is a key element for designing smarter operational controllers. It provides the necessary instructions and parameters to allow the controller to make decisions using different operational control logic in different situations. The control policy is presented as a standard object enabling flexible decision-making in operational control, which is necessary to realize a specific behavior of the base system.

4.2 Controller Behavior

A sequence diagram is shown in Figure 4 to depict the dynamic behavior of the operational controller, which shows how the proposed *ControlMap* and *ControlPolicy* objects integrate within the existing controller architecture. As soon as an event triggers the *EventDirector*, the *ControlMap* searches for the proper control policy, and this information is passed to the *AnalysisFormer*. The *AnalysisFormer* checks the control policy to detect which input parameters are required, and then connects to the *ModelQuerier* to retrieve such parameters from system model. Finally, the solver receives both the control policy and its input parameters and can execute the control policy solve method to obtain the decision variables.

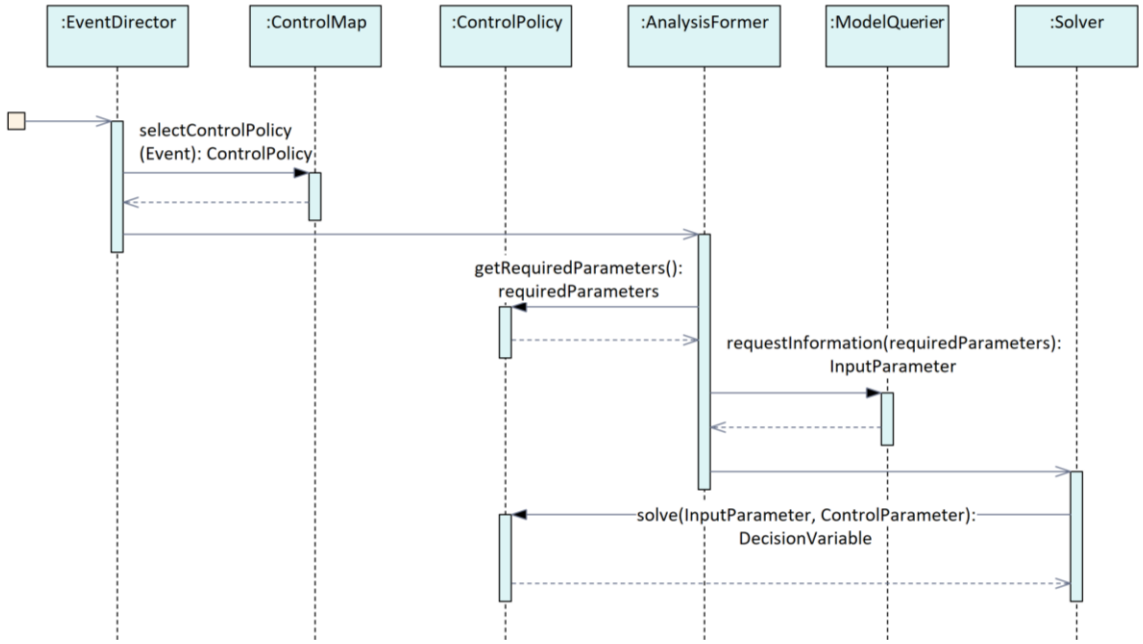


Figure 4: Controller behavior.

5 IMPLEMENTING SMART CONTROL IN AN ANYLOGIC™ SIMULATOR

The proposed framework for implementing operational controllers in DES was applied to the logistics hub presented in section 3 as demonstration platform. As stated, controllers at cell and at hub levels are responsible for sequencing robot tasks and for changing cell states, respectively.

5.1 Technical Aspects of Implementation

The operational controllers are implemented as agents and located where their control decisions are required (i.e., within either the logistics hub or the shuffle cell). The controller agents' behavior is designed using AnyLogic™ finite state machines. In this way, the controller stays in "idle" state until it is triggered by some event to switch to a "working" state, where control functions are executed to set decision variables and authorize tasks to be executed. The generic controller implementation is illustrated in Figure 5.

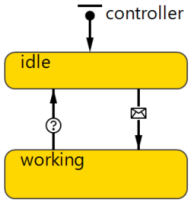


Figure 5: Cell controller state machine in AnyLogic™.

Communication is realized using messages, in such a way that specific control actions can be triggered, and allowing for the same controller to handle multiple control decisions. Conversely, both Control Map and Control Policy are implemented in AnyLogic™ using *Action charts*, which visually represent portions of code and can be called as functions.

On the reception of a message (i.e., when an event occurs), the Control Map is queried to determine which Control Policy should be triggered, if any. The typical structure of the Control Map consists of multiple if-else blocks which allow the implementation of its Boolean logic. While designing a system, the complexity of the control map increases progressively to reflect the different aspects being modeled. Likewise, control policies can be designed to execute operational control decision-making processes.

As system complexity increases during the design process, model execution requires handling a growing number of events. For this reason, the *ControlMap* is iteratively refined to respond according to designer expectations. The *ControlMap* implemented within the cell controller is illustrated in Figure 6. Throughout the system design process, the *ControlMap* is stepwise defined allowing to refine the behavior of the controlled element.

Different colors correspond to successive steps in the design. First, the cell is only triggered to start shuffling as a rack arrives (a). Then, the controller switches between the "idle" and "working" states of the controller according to the cell content (b). With a successive iteration, the controller triggers the sequencing of the cell tasks as soon as a new totebot enters (c) or leaves (d) the cell.

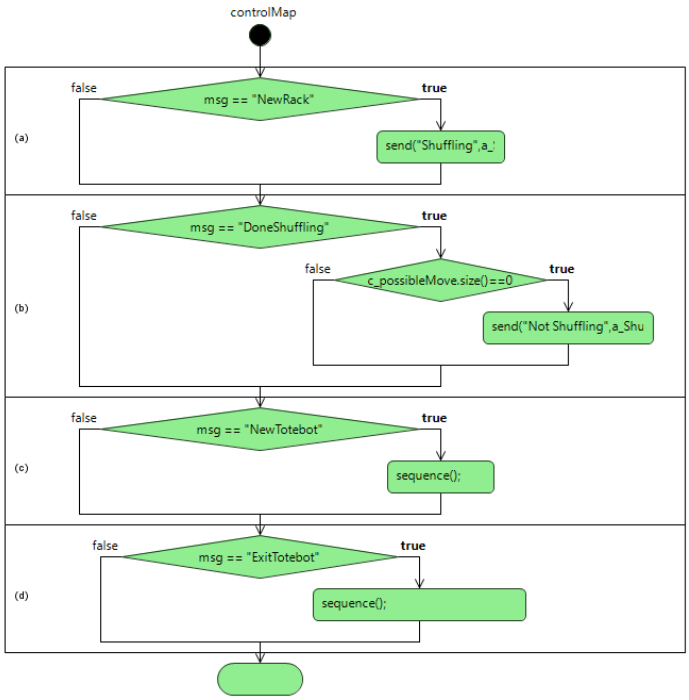


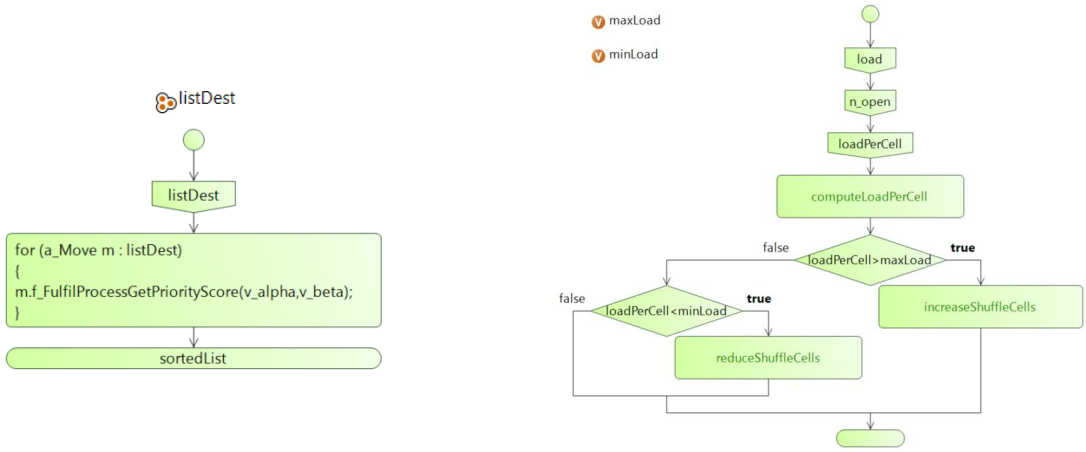
Figure 6: Control map.

In order to move towards the use of operational controllers during system operation, the control policies shall be further structured by collecting and exposing some control parameters. As explained in section 4.1, such parameters permit adapting the behavior of the controller in a flexible way, allowing to:

- Update the DT as changes are made in the operational control of the system;
- Conduct what-if analyses to evaluate changes in the way the system is controlled, allowing to react promptly to disruption or external disturbances.

Action charts can be customized to fully encapsulate information and behavior required for operational control decision-making, and to visually represent the control logic.

Coherently with the structure shown in Figure 2, the controller includes several Control Policy objects, which are stored in the controller agent. When the controller is triggered by a message received, it will run the control strategies gathering the required input parameters from the system prior to execute the control function. Figure 7 shows how control policies were implemented in AnyLogic™.



(a) (b)

Figure 7: Control policy implementations in AnyLogic™.

5.2 Computational Results

The approach to modeling operational control described in section 4 has been applied to the use case outlined in section 3 by implementing it in the logistics hub simulator from Montreuil et al. (2021). The experiments described here illustrate the capability to rapidly change control policies and evaluate their impact on the productivity and the service rate of the hub.

The number of totes departed from the hub is used to evaluate system throughput (TH). The average lead time is also reported. The service rate accounts for the percentage of totes delivered on time each hour. For these metrics, the standard deviation (STD) is also provided.

The first set of experiments aims to graphically present the effect of adding control triggers to the shuffle cell controller. As explained in section 5.1, the *ControlMap* is refined during the design phase. Simulation results shown in Figure 8 allow a graphical comparison to verify how this is impacting the behavior of the controlled system. In the first step of the controller design, the cell is not controlled correctly as its state is not switched to "Idle" after task completion (a), which is overcome in the second design iteration (b), which allows reacting to the "Done Shuffling" event. From the third iteration on, the controller correctly sequences the shuffling tasks again each time a totebot enters the cell. Finally, last iteration allows to perform this operation also when a totebot leaves the cell.

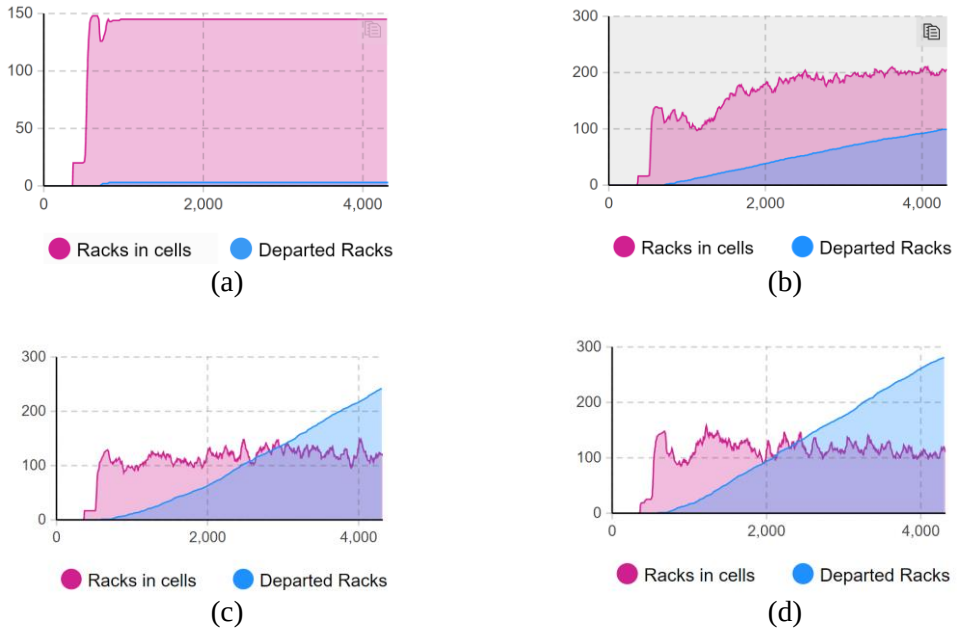


Figure 8: Results of experiment 1.

The second set of experiments aims at showing how the proposed implementation of operational controllers allows what-if analysis during system operation. In fact, three different dispatching rules are tested within the shuffle cells, which show the changes in the system behavior:

- Earliest Due Date (EDD). The due date considered is the departure time of the truck to which each tote is assigned. The shuffle cell robot always executes the move task related to the tote whose truck has the earliest departure time.
- Adjusted Earliest Due Date (A-EDD). Computed as the difference between the due date and the release time, which is the time the tote arrived at the hub. This dispatching rule is supposed to favor the consolidation of racks, increasing the effort put on those racks whose consolidation process has been started earlier, and thus the probability of completion could be higher.
- First-In-First-Out (FIFO). The tasks are executed in the same order as they are totes received in the shuffle cell.

The control policy only computes the scores for each of the shuffle robot's possible moves using the priority described by the dispatching rule, and sort those moves accordingly.

Table 1: Results from second experiment.

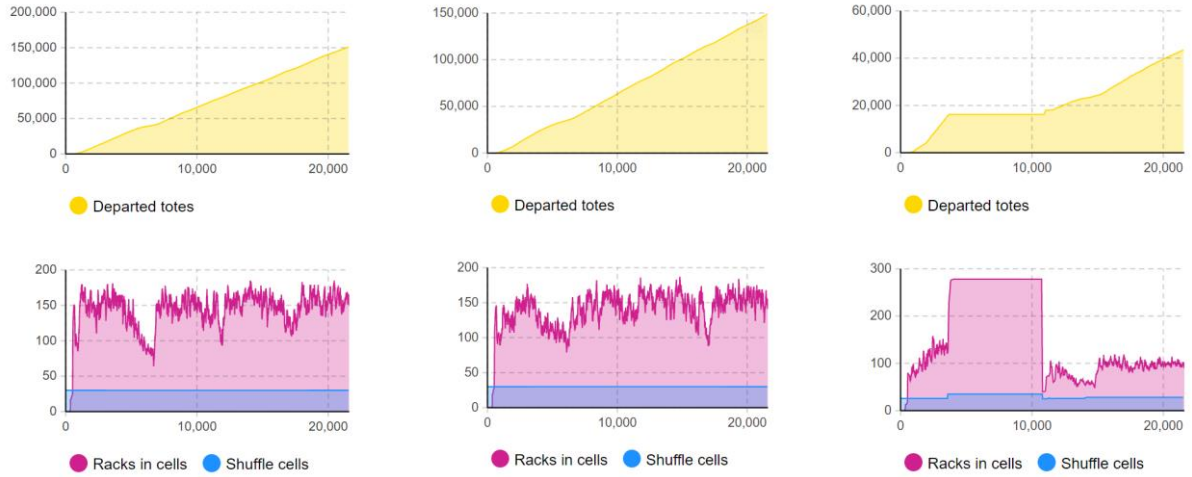
Exp.	Parameter	Service rate		Lead time		TH [totes/min]
	Dispatching rule	Mean [%]	StdDev [%]	Mean [minutes]	StdDev [minutes]	
2a	EDD	72.55	21.74	45.34	32.76	420.24
2b	A-EDD	73.61	20.8	46.20	32.01	414.93
2c	FIFO	71.53	22.45	57.02	59.12	398.95

Note that each of the three policies is best on one of the three criteria, which supports the notion that in operation, the controller should be "smart enough" to be able switch among policies based on current conditions. Figure 9 reports the results of each experiment. The top charts show the number of totes shipped over time, while the bottom charts show the number of open cells and the number of racks in the cells.

The second set of experiments shows how the model supports decision-making during system operations. In fact, it allows to:

- Evaluate system performance varying the number of open cells, i.e., the cell currently active;
- Define and tune specific rules to open and close cells based on the current workload on the system. In this case, the control policy computes the load on shuffle cells based on the average number of racks per cell, according to equation (1).

$$loadPerCell = \frac{1}{N-1}\sum_{0}^{N-1} RacksInCells_i \quad (1)$$



(a) (b) (c)

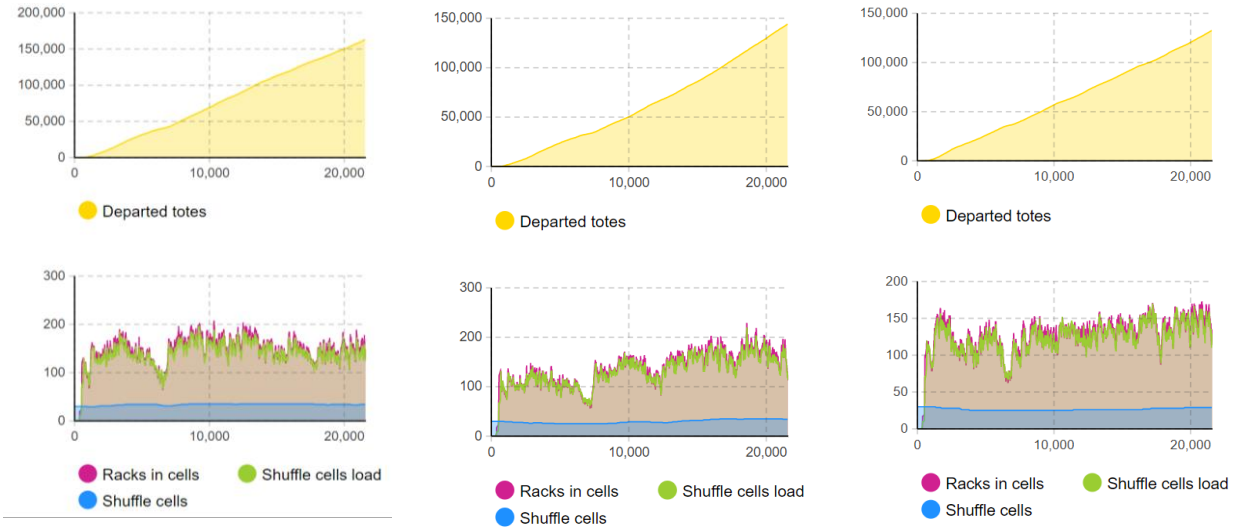
Figure 9: Results of experiment 2.

The control policy for opening or closing shuffle cells has two parameters, namely *maxLoad* and *minLoad*. When the *loadPerCell* falls below *minload* a cell is closed and when it rises above *maxLoad* a new cell is opened. The policy works like a control chart, opening and closing shuffle cells to force the *loadPerCell* value in the interval [*minLoad*; *maxLoad*]. During system operation, these parameters can be set to effectively manage the system workload. Results are summarized in Table 2. Note that increasing the *maxLoad* trigger results in a lower throughput, which is intuitively correct, because it would delay the opening of new shuffle cells.

Table 2: Results from third experiment.

Exp.	Parameters		Service rate		Lead time		TH [totes/min]
	minLoad	*maxLoad*	Mean [%]	StdDev [%]	Mean [minutes]	StdDev [minutes]	
3a	3.5	4.5	74.81	20.07	36.73	26.30	454.73
3b	4	5	60.67	37.71	40.96	28.94	401.76
3c	4	5.5	60.10	37.66	65.99	41.04	369.60

Figure 10 reports the results of each experiment. The top charts show the number of totes shipped over time, while the bottom charts show the number of open cells, the number of racks in the cells, and the load per cell computed according to equation (1). In a system design scenario, many more experiments would be run to refine the setting of key parameters and to evaluate alternative control policies. What these two sets of experiments indicate is that the proposed functional architecture and controller implementations support the development and use of a high-fidelity simulation, or digital twin, in the design process.



(a) (b) (c)

Figure 10: Results of experiment 3.

6 CONCLUSIONS AND FUTURE RESEARCH

This paper has presented a novel approach to designing and implementing operational controllers in DELS. This is based on a generic approach to control DELS systems by specifying how the system reacts to a range of different events with a variety of control strategies. The approach offers the possibility to concurrently design the DELS and its control system, allowing increased flexibility in the use of the DT.

This paper has presented a novel approach to designing and implementing operational controllers in DELS. This is based on a generic approach to control DELS systems by specifying how the system reacts to a range of different events with a variety of control strategies. The approach offers the possibility to concurrently design the DELS and its control system, allowing increased flexibility in the use of the DT, while significantly decoupling the implementation of the control model from the implementation of the base system model.

This work has therefore important implications for the development of DES models to support DELS design and operation. By providing a standardized structure for control policies, this study provides a new way to explicitly integrate operational control decision-making into DES models. This enables controllers to be used in a more flexible way, without requiring the redesign of controllers as the system evolves and changes within both its design and operation phases. This could facilitate the exploration of different control policies during the design of a new system, and the change of existing ones while the system is in operation.

Importantly, the use of control policies in DES models may help to better align model control with the control of real systems they represent, with the possibility to increase model fidelity.

Looking to the future, it will be important to investigate the potential impact of this modeling technique on the application of DTs in system operation. The fact that experimentation with control policies is easily supported in the design scenario implies that such experimentation might also be employed as an off-line adjunct to real-time operational controllers. The off-line experimentation could explore the impacts of current decisions about control policies and parameters on the future performance of the controlled system.

The definition of a standard way to model operational control policies will encourage the use of DT for addressing operational control problems, thanks to the modularity of the control policies, which may eventually allow the online self-optimization of controllers.

In summary, the approach presented in this paper offers a promising new avenue for the development of more flexible and efficient controllers in DES models of DELS, since this work can help to streamline

the design process and improve the accuracy of modeling and simulation efforts. A final note: because the operational controller approach presented here is event-driven, there is a clear migration path to industrial implementation. This would be a major step forward in bringing the control of the "real twin" and the control of the "digital twin" closer together in the DELS domain.

REFERENCES

Babalou S., W. Bao, A. Barenji, B. Montreuil, L. McGinnis, S. Buckley (2021). "Modular and Mobile Design of Hyperconnected Parcel Logistics Hub". In *Proceeding of IPIC 2021 International Physical Internet Conference*, 10 pp.

Cui, Yiqian, Junyou Shi, and Zili Wang. 2015. "Discrete Event Logistics Systems (DELS) Simulation Modeling Incorporating Two-Step Remaining Useful Life (RUL) Estimation". *Computers in Industry* 72: 68–81.

Ehmke, Jan Fabian, Daniel Großhans, Dirk Christian Mattfeld, and L. Douglas Smith. 2011. "Interactive Analysis of Discrete-Event Logistics Systems with Support of a Data Warehouse". *Computers in Industry* 62 (6): 578–86.

McGinnis, L. F. (2010). "The Future of Modeling in Material Handling Systems". In *11th International Material Handling Research Colloquium* (pp. 261-274). Charlotte, NC: MHI.

McGinnis, L. F. (2019). Formalizing ISA-95 Level 3 Control with Smart Manufacturing System Models. NIST Interagency/Internal Report.

McGinnis, Leon, Buckley, S., and Barenji, A. V. (2021). "Designing and Implementing Operational Controllers for a Robotic Tote Consolidation Cell Simulation". In *Proceedings - Winter Simulation Conference*. Vol. 2021-December.

Montreuil, B., McGinnis, L., Buckley, S., Babalou, S., Bao, W., and Beranji, A. (2021). "Physical Internet Induced Parcel Logistics Hub Innovation". *International Physical Internet Conference*.

McGinnis, L. F., Huang, E. and Wu, K. (2006). "Systems Engineering and Design of High-Tech Factories". In *Proceedings - Winter Simulation Conference*, 1880–86.

Sprock, T. A. (2016). A Metamodel of Operational Control for Discrete Event Logistics Systems. Retrieved from https://smartech.gatech.edu/handle/1853/54946, accessed 10th March.

Sprock, T., Bock, C., and McGinnis, L. F. (2019). "Survey and classification of operational control problems in discrete event logistics systems (DELS)". *International Journal of Production Research*, 5215-5238.

Sprock, T., and McGinnis, L. F. (2015). "A Conceptual Model for Operational Control in Smart Manufacturing Systems". *15th IFAC Symposium on Information Control Problems in Manufacturing*. 48, pp. 1865-1869. IFAC-PapersOnline.

Sprock, T., Thiers, G., McGinnis, L., and Bock, C. (2019). Theory of Discrete Event Logistics Systems (DELS) Specification. NIST Interagency/Internal Report. Retrieved from https://doi.org/10.6028/NIST.IR.8262 , accessed 18th February.

Thiers, George, and Leon McGinnis. (2011). "Logistics Systems Modeling and Simulation". In *Proceedings - Winter Simulation Conference*, 1531–41.

AUTHOR BIOGRAPHIES

LORENZO RAGAZZINI is a postdoctoral researcher at the Department of Management, Economics and Industrial Engineering. He works as a researcher in the Manufacturing Group of the School of Management at Politecnico di Milano. His main research interests include decision-making frameworks supported by Digital Twin simulation and data-driven intelligence. His email address is lorenzo.ragazzini@polimi.it.

LEON MCGINNIS is Professor Emeritus at Georgia Tech, where he continues to teach and do research related to DELS operational control and design. His email address is leon.mcginnis@isye.gatech.edu.

ELISA NEGRI is Senior Assistant Professor in the Manufacturing Group of the Politecnico di Milano at the Department of Management, Economics and Industrial Engineering. Her research focuses on the impact on production planning and control of smart manufacturing and the digital technologies of the Industry 4.0. Her email address is elisa.negri@polimi.it.

MARCO MACCHI is Full Professor, currently teaching Industrial Technologies, Asset Lifecycle Management and Smart Manufacturing Lab at Politecnico di Milano, also acting as Director of the Master programme meGMI by Polimi Graduate School of Management and the School of Management of Università degli Studi di Bergamo. He is chair of the IFAC Technical Committee TC5.1 Manufacturing Plant Control, Associate Editor of Journal of Intelligent Manufacturing, Editorial board member of the International Journal of Production Planning & Control: The Management of Operations. His email address is marco.macchi@polimi.it.