Proceedings of the 2020 Winter Simulation Conference
K.-H. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing, eds.

AN ANALYSIS-AGNOSTIC SYSTEM MODEL OF THE INTEL MINIFAB

Leon F. McGinnis

School of Industrial and Systems Engineering
The Georgia Institute of Technology
Atlanta, GA 30332-015, USA

ABSTRACT

What if, instead of trying to model production systems using a simulation language, we first formally specified them by creating an analysis-agnostic system model (AASM) which could be developed collaboratively with the stakeholders and then be used as a simulation model requirements document, the benchmark for simulation model verification and a valuable asset for validation? This paper demonstrates an approach for creating an AASM for a wafer fab, using as the demonstration vehicle a famously simple yet intriguingly complex case study, the Intel Minifab case developed by Karl Kempf 25 years ago.

1 INTRODUCTION

Engineering design automation (EDA) is a key factor, perhaps the single most important factor, in the creation of modern engineering artifacts like the Boeing 787, JPL's Europa Clipper, AMD's Zen 2 based Epyc Rome, and many others. A fundamental requirement in every example of EDA is an underlying analysis-agnostic representation of the system being designed, in its own terms. This representation, whether of a class of aircraft, a space mission or a system-on-a-chip integrated circuit, always conforms to some underlying reference model of the class of systems of interest. The designers are creating a description of the elements of the system being designed, i.e., creating the AASM that describes the system; subsequent integrated analysis models predict how it will perform.

The design of wafer fabs is no less challenging than the design of the devices fabricated in wafer fabs. Wafer fab design requires decisions about capacity, technology, operational planning and scheduling, staffing, maintenance, and many more issues. As with the devices to be produced, designing the production system requires the ability to assess the trade-offs involved in making these decisions. Many different analysis models are used in those assessments, and simulation is an important one, but not the only one.

Suppose there was true EDA for designing wafer fabs, i.e., making decisions about resource portfolios, resource arrangements, planning strategies, and operational control decisions. This EDA would necessarily give the designer access to analysis models that support design decision-making, including simulation models, just as is true with the EDA for designing produced systems. In order to realize wafer fab EDA, it is clear there must be a wafer fab-specific AASM, if only to ensure that the multiple analyses are mutually consistent in assumptions about the designed system. Definition of the wafer fab, i.e., its AASM would precede the creation of high-fidelity simulation models conforming to the AASM. It also would provide the necessary common specification needed to integrated decisions relating to resources, capacity, product mix, inventory, and other critical elements of modern wafer fab design.

The motivation for this paper is the critical need for an AASM approach for representing production systems in general and wafer fabs in particular. What is presented here is a demonstration of one possible approach to creating wafer fab models that are, in fact, analysis agnostic. It is not, by any stretch, the complete solution or the ready-to-deploy AASM for all wafer fabs. However, it does demonstrate that wafer fab AASM is not a pie-in-the-sky idea. There are tools and methodologies available today to support the development of wafer fab AASM. The model presented here was created using OMG SysML™ as the

modeling language and the MagicDraw editor from NoMagic, Inc. Shown here are only key parts of the model, but the complete model is available from the author on request.

The following section briefly describes the target case study, the Intel Minifab (IMF) from (Kempf 1994). Section 3 briefly describes the conceptual framework for the AASM. The application of the modeling framework to develop an AASM for the IMF is described in section 4 and section 5 provides some concluding remarks.

2 THE INTEL MINIFAB CASE

The original "Intel Five-Machine Six Step Mini-Fab Description" (Kempf 1994) describes a simple 3 product, 3 cell, 3 machine types, and 6 operation types case that incorporates many of the complications found in real wafer fabs. There is re-entrant flow, preventive maintenance and breakdown repair (both performed by a single maintenance technician, who must take breaks and attend meetings), machine setups, batching, and machine loading and unloading (which must be performed by one of two production operators, each of whom has a specific range of cells to service and must take breaks and attend meetings). There are three cells, corresponding to types of operations, and each cell has a stocker from which lots are retrieved to be loaded onto machines and to which lots are returned after processing. There is an automated transport system that moves lots from the starts stocker to cell 1, among the cells to complete the process route, and to the outs stocker. Deterministic capacity analysis of each resource type—ignoring setups and queuing delays—indicates sufficient capacity to meet the required throughput rate. Figure 1 illustrates the logical architecture of the IMF.

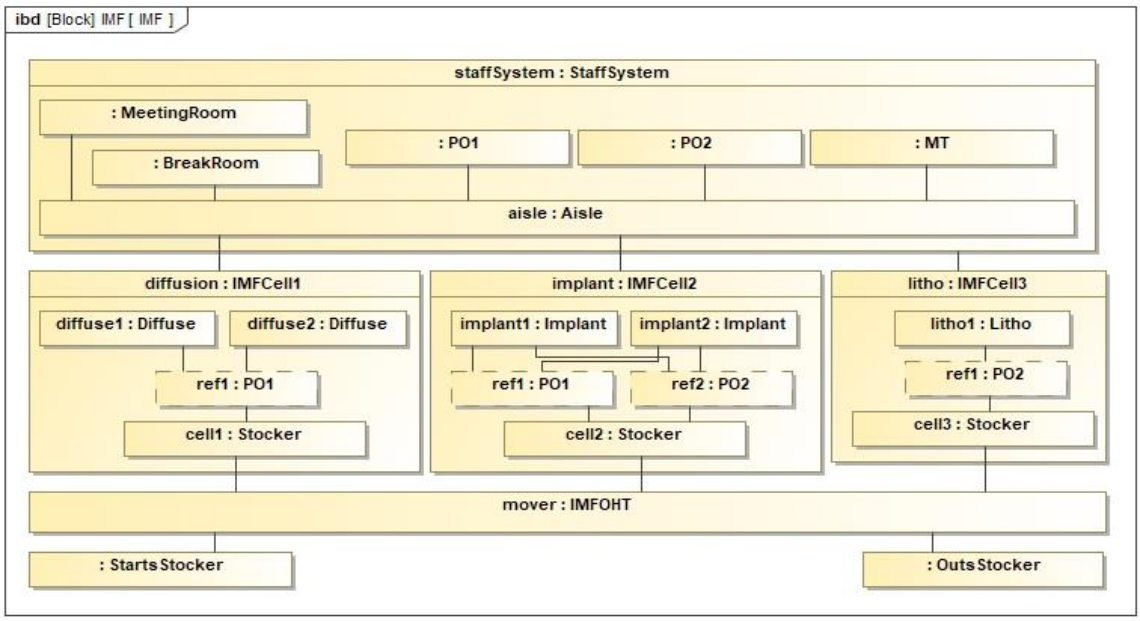


Figure 1: IMF Logical Structure.

Figure 1 presents some graphical notation that is important. While *PO1* and *PO2* are parts of the *StaffSystem*, they are represented in the three cells as dashed boxes because they can be assigned temporarily to the cells and are the means by which lots move between stockers and machines.

(Kempf 1994) also models the states and state transition for the lots, machines, operators, technicians, and transporter, and identifies the operational control decisions required. In his formulation, the only indication of a control system structure is that the operators and technicians decide their sequence of operations, including switching between cells and when to take breaks. The stated performance

requirement for the IMF is the number of starts per week for each product, and the identified metrics are: maximize outs, minimize variability of outs, minimize throughput time, minimize variability of throughput time, minimize work-in-progress and maximize utilization of lots, machines, operators, technicians and transporter. No approach for managing trade-offs between these criteria is suggested.

A large number of publications have used the IMF as a testbed for evaluating proposed scheduling rules. To the best of the author's knowledge, none of these evaluations has incorporated all features of the IMF.

3 CONCEPTUAL FRAMEWORK FOR AN AASM OF THE INTEL MINIFAB

Since 2006, I've used the phrase "discrete event logistics system," or DELS, to describe the class of systems into which the IMF falls (McGinnis *et al* 2006, Nazzal and McGinnis 2006, Lin and McGinnis 2017) and DELS have been a focus of research in the Keck Virtual Factory Lab at Georgia Tech (see, e.g., Thiers 2015 and Sprock 2016). Often that prior work has addressed the possibility of generating simulation models from an AASM, but this paper addresses the more fundamental issue of the sufficiency of a DELS-based AASM for specifying the system of interest.

A DELS is a network of resources through which discrete jobs flow and are transformed by operations performed by the resources and having discrete start and end times. The essential semantics of DELS are shown in Figure 2. The terms used are reasonably self-defining, although one should note carefully that resources can contain "member" resources, which implies that a process associated with a resource may consist of multiple sub-processes that are executed by *memberResources*. Note also that executing a process—which is the capability of some resource—requires an authorization, shown as a *Task*; a controller "tasks" a resource to execute a process. The fundamental DELS concept of operation is that a DELS receives tasks, or requests for a product or service; its controller decides whether or not to accept the task, and if it accepts, translates the task into subtasks that can be executed by its *memberResources* (some of which may themselves be DELS) or offered to other DELS which are not *memberResources* (see, e.g., Sprock *et al* 2019). The specification of a particular task is context-dependent and can be as broad as "produce *u* units of product *p* by deadline *d*," or as specific as "execute machine program *m*".

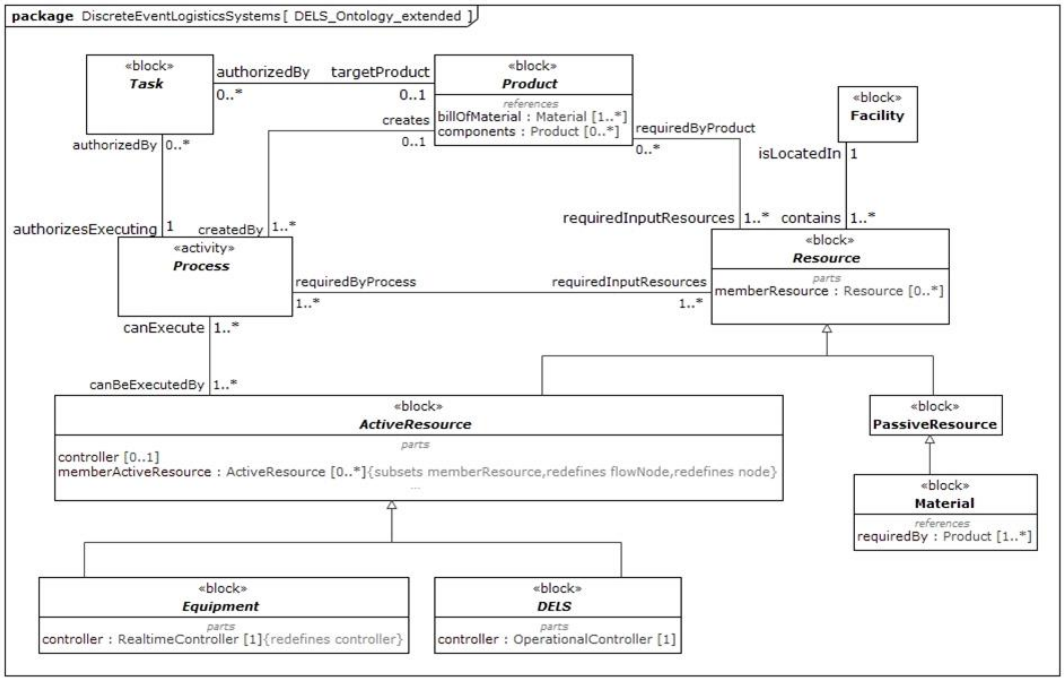


Figure 2: Essential DELS Semantics.

There are two fundamental requirements for an effective DELS AASM: (1) it must explain, at a sufficient level of fidelity, what can be observed in the DELS, which includes the resources, processes and products; and (2) it must explain what operational control decisions are required and how they are made. It is not necessary to explain how the individual product transformations are performed as long as the time required can be predicted. Thus, process execution is explained adequately by start time, end time, and product state change. The proposed approach for organizing a DELS AASM is product, resource, process and control.

For further details on the development of the DELS reference model, see (Sprock *et al* 2019). For an example of modeling operations management in a central fill pharmacy, see (McGinnis 2019).

4 MODELING THE IMF

The IMF is not a "real" fab, and one indicator is the context. As defined in (Kempf 1994) the IMF has a fixed constant lot release rate for its three products.

4.1 Product

The IMF *ProductSet* contains *ProductA*, *ProductB* and a test wafer, *TW*. However, what moves through the IMF is a container of wafers, modelled here as a front-opening unified pod, or FOUP which will have properties that include its current *ProductID*, *ReleaseDate*, *DueDate* and perhaps others. For the IMF the processing requirements for each of the three products can be described as a sequence of actions or process steps. These steps are technical definitions of the transformation that must be achieved and must be translated into process specification for any resource capable of achieving the transformation; different resources capable of the same transformation may have different instructions and performance metric values. The only process steps that actually depend on the wafer type are steps 3 and 6 which are described as lithography steps and require a setup specific to the part type and step being run.

4.2 IMF Resource Model

As shown in Figure 3(a), IMF resources are seven components or subsystems, which are themselves DELS, with their own resources and operational controls. Collectively, these resources provide the process capabilities of the IMF. Since these owned resources—the subsystems—also are DELS, they need to be described in terms of product, resource, process and control, which will be done in subsequent sections.

Each resource must publish a list of the processes for which it is capable, i.e., processes it can execute to produce a product in the *ProductSet*; these are its relevant behaviors.

4.2.1 Stockers and *IMFMover*

The IMF has starts and outs stockers, and each cell has a stocker. Figure 3(b) illustrates the generic stocker model, with value properties for FOUP capacity and FOUP inventory, and operations that correspond to putaway and retrieval from/to an automated transport (M) or a production operator (P). Note that operations involving the production operator allow for batches, and the assumption is that all FOUPs in a batch are either retrieved or stored consecutively. Clearly, the batch size would be one when the production operator is servicing a machine with single FOUP capacity. The implementation of the get/put operations will include publishing corresponding event information. In the IMF, each specialization of *Stocker* can have a different value for inventory capacity.

The *IMFMover* has a single exposed behavior, which is *move(FOUPID, source, destination)*. If deadheading is required, its controller must determine the path from its current location to *source* as well as from *source* to *destination*. Also, the *IMFMover* must interact with the five stockers to execute its behavior.

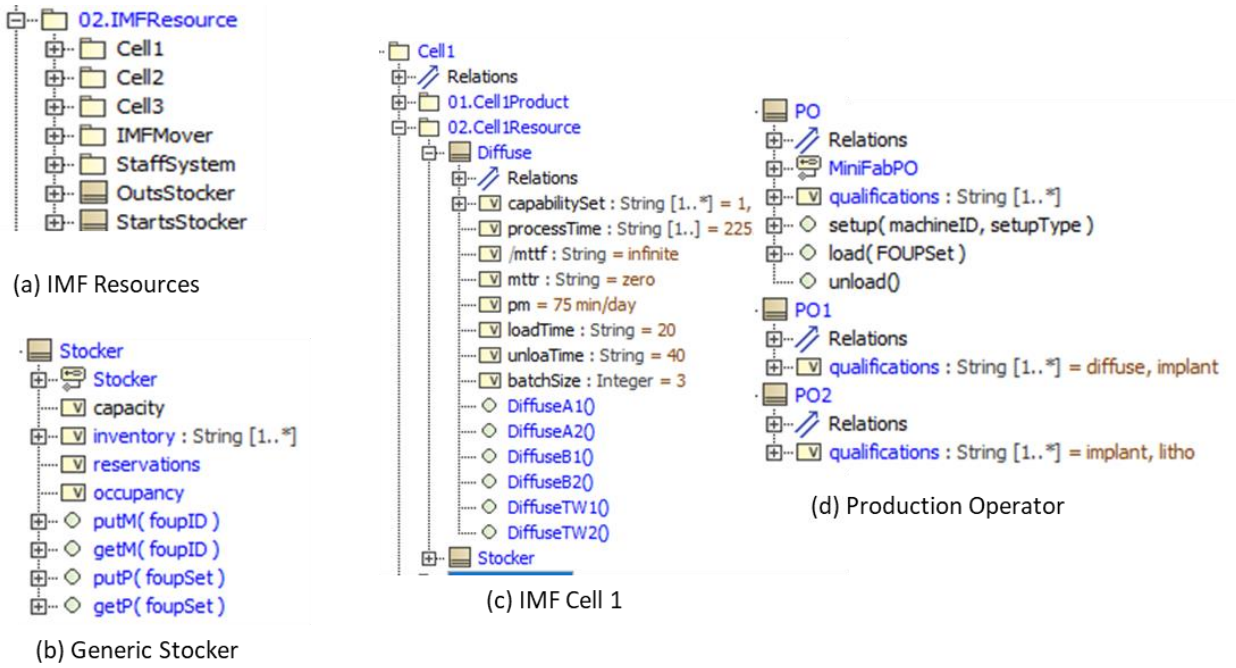


Figure 3: IMF Resource Models.

4.3 IMF Process

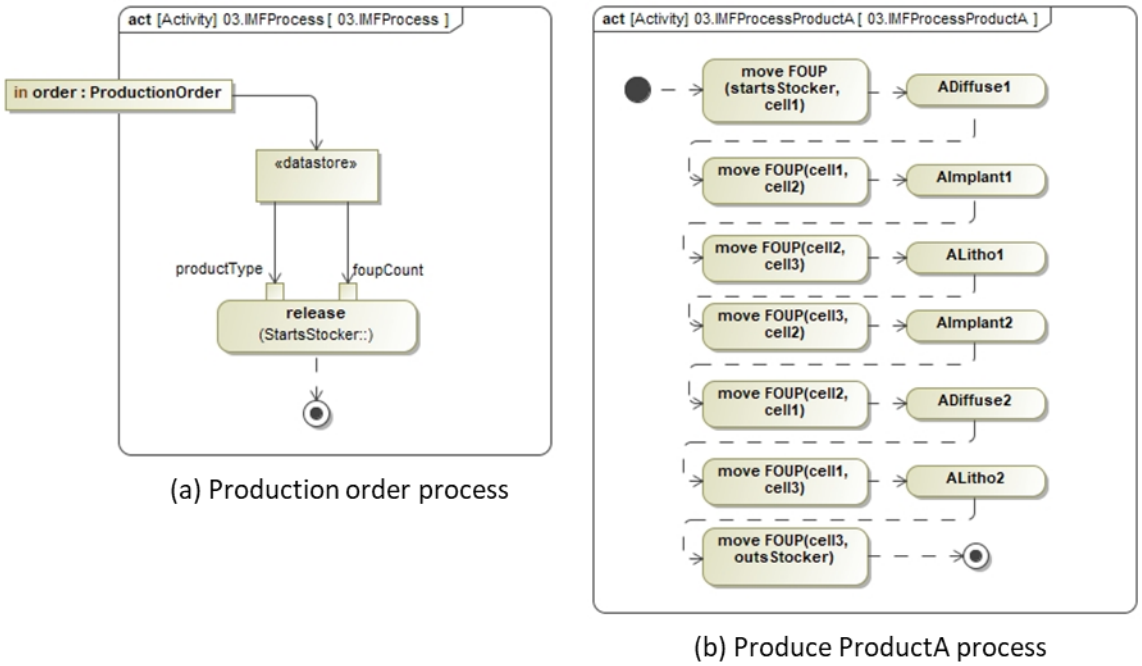
DELS will expose physical transaction processes corresponding to input or output of physical product. For example, if the IMF received FOUPs loaded with bare wafers, there would be a process for receiving them. Similarly, if the IMF delivered FOUPs loaded with finished wafers, there would be corresponding processes. For the IMF as presented in (Kempf 1994) the starts and outs stockers in effect "hide" these processes.

Our model exposes only one process of the IMF, the lot ordering process, as illustrated in Figure 4(a). Once per week, the IMF will receive production orders for all required products and will release the corresponding number of FOUPs to the *StartsStocker*. This process is modeled as an activity *IMFProcess* with signature (*ProductionOrder*) and is exposed as a call behavior action of the IMF itself. Here *ProductionOrder* has two value properties, *productType*, and *foupCount*. One can easily imagine that an alternative formulation of the IMF model might require an exposed process for delivering product, e.g., an activity with signature *Deliver*(*ProductionOrder*, *dueDate*). This would lead to a much more difficult case, requiring the ability to plan the release dates, specifying how to deal with tardy completions, and many other complications.

DELS do not, in general, expose their internal resource processes, i.e., the processes that can transform materials. Rather, their capabilities are exposed as calls to their control processes. Equipment resources may be modeled in the same way, but since they make no operational control decisions it may be simpler to model their process capabilities directly as called behavior actions.

The IMF has the capability to produce products A, B and TW. Although not exposed, that capability can be modeled as an activity diagram, illustrated in Figure 4(b) for *ProductA* which represents all processes required to support the flow of FOUPs through the IMF. A similar activity would describe the process capabilities for *ProductB* and *TW*. Note that since *Cell1* is a DELS, how it performs *ADiffuse1* is not exposed to *IMF*. While these activity models define precisely how *IMF* produces products, they do not explain how operations (the moves and the cell operations) are scheduled and sequenced.

Each of the process steps in Figure 4(b) may be elaborated into multiple steps. For example, *Cell1* is capable of executing both *ADiffuse1* and *ADiffuse2*. These two processes also involve multiple steps, including retrieving a batch of FOUPs from the *Cell1Stocker*, transporting them to the assigned machine, loading the machine, running the diffusion process, and then unloading the machine and returning the FOUPs to the *Cell1Stocker*. In general, the specification of process steps must be structured in a manner that conforms to the control architecture of the system that will execute the process steps and must be augmented with the required move and store processes.



(a) Production order process

(b) Produce ProductA process

Figure 4: IMF Process Models.

4.4 IMF Control

(Kempf 1994) does not specify a control system architecture, so modeling IMF control implies *designing* a control system. In fact, this is exactly the same situation in which simulation modelers often find themselves, and they may make implicit assumptions about what can and cannot be done to control operations. Creating the AASM forces the modeler to be explicit. It would be helpful to have some principles to guide such design decisions. One such principle might be called ***control encapsulation***.

In the logical structure shown in Figure 1, control encapsulation would mean that each DELS (*IMF*, *StaffSystem*, *StartsStocker*, *OutsStocker*, *IMFMover* and the three cells) has its own controller which manages its external interactions and internal operations, with objectives and constraints provided by its owning DELS, or its context in the case of the IMF. The IMF itself manages the product flows between its owned DELS, consisting of FOUPs moved by the *IMFMover* and process steps executed by the cells (each tasked by *IMFController*). POs and MT move (as tasked by the *StaffSystemController*) among cells to provide service to the cells, and within the *StaffSystem* to meet requirements for meetings and breaks. Similarly, the stockers, the cells and the *IMFMover* manage their own internal operations, based on objectives provided by the *IMFController*. Note the POs and MT are themselves DELS, and can manage their movement within *IMF*

Suppose we want to model the class of systems, for which the instance described in (Kempf 1994) is one example. We might want to experiment with different cell configurations, more than one FOUP mover,

different product mixes, different products and even different process plans. If that is the goal, then we need an abstract definition of the controller.

Functional decision requirements for a DELS controller identified by (Sprock 2016) include *admission* (accept a task or not), *assignment* (assigning a task to a resource), *sequencing* (order of assigned task execution), *routing* (deciding what happens next), and *resource configuration* (essentially, changing setup). Consider each of these in turn for the IMF controller in the context of controlling the flow of FOUPS in the IMF.

The *IMFController* accepts all *ProductionOrder*s, because the *StartsStocker* has unlimited capacity. Each new lot released to the *StartsStocker* must eventually result in a set of tasks corresponding to the processes identified in Figure 4(b). Constructing these tasks requires the *IMFController* to have access to five data sources: (1) product set for which IMF is capable; (2) product-specific process plans; (3) the set of capable resources associated with each step of the process plan; (4) any constraints on process step to resource assignment; and (5) the resource-to-resource connectivity.

The *ProductSet* and the form of the product-specific process plans are described in section 4.1. Each resource has a *capableProcessList* identifying those processes from the process plans for which the resource is capable. For example, *IMFCell1* is capable of executing *Adiffuse1* and *ADiffuse2* for *ProductA*, as well as the corresponding diffusion processes for *ProductB* and *TW*. In the IMF, there are no process step to resource assignment constraints, other than those explicitly defined in the process plans. The resource-to-resource connectivity is explicitly defined in Figure 1. Thus, the information is available to construct the tasks corresponding to the process illustrated in Figure 4(b).

Exactly how tasks are constructed is an implementation issue, but the structure of the tasks must include the following information:

(taskID, resourceID, taskType, parameterSet)

where resourceID is the resource that will execute the task, taskType identifies the capable process to be executed, and parameterSet is the set of parameters that the resource will need in order to execute taskType. For the process illustrated in Figure 4(b), the first two tasks would be:

(taskID, *IMFMover*, move, (*FOUPID*, *StartsStocker*, *Cell1Stocker*))

(taskID, *IMFCell1*, process, (*FOUPID*, *ADiffuse1*))

Once a task is created, it is in one of five states:

- Pending: state when initially created
- Available: all predecessor tasks are complete
- Issued: the task has been assigned to a capable resource
- Completed: the assigned resource reports the task as finished
- Failed: the assigned resource reports the task as failed

Suppose the *IMFMover* and the three cells have the capabilities to execute all the process steps required to produce the *ProductSet*. The remaining control issue to resolve for the *IMFController* is how the corresponding tasks are managed, i.e., how the *IMFController* moves the task from pending to available to issued. This is, in fact, the heart of the control system design problem for the IMF. The objective here is not to elaborate a control theory or a control system design methodology, but rather to demonstrate that a control design can be articulated in a manner that supports subsequent analyses.

Events determine when a task changes state. For example, a move task switches from pending to available when its preceding process task is complete. A process task switches from pending to available when the target FOUP is in the cell's stocker. In general, the completion of any issued task is an event that may change the status of pending or available tasks as well as the state of the assigned resource for the completed task.

Events also determine when the *IMFController* has the opportunity to make a task assignment decision. Any event that triggers a change in task state also triggers a decision opportunity. In addition, events that change the state of an *ownedResource* can trigger a decision opportunity. Examples might include expiration of timers for PM, or the occurrence of a tool failure.

In the IMF context, whenever the *IMFController* receives a task-related message, it will update status for all affected tasks, update status for all *ownedResource*s and initiate a decision cycle. The decision cycle will consist of (1) identifying the set of available tasks that are not blocked because no capable resource is available; (2) if the set is non-empty, selecting the best task to assign (or no task); (3) assigning the task by publishing the task message; and (4) repeating until there are no more tasks that can be assigned. Figure 5 provides a pseudo-code model for the **controller decision cycle**.

This approach to modeling control is completely data-driven. A generic controller infrastructure based on the described data can accommodate changing the number of products, provided, of course, the associate process plans are specified, and the resource capabilities are updated. Similarly, changing the number of cells or the number of steps in the process plan can be accommodated.

The general, re-usable principles are the product model, which specifies technical product transformations required; the resource model, which specifies the capabilities of individual resources and their connectivity; and the elaboration of the product-oriented process requirements into a production-oriented process requirement that includes movement and storage. This information enables the computational creation and state classification of tasks. By maintaining a database of tasks and task states, and resources and resource states, the decision cycle is independent of the specific set of products, processes or resources.

Within the decision cycle of Figure 5, the actual decision-making itself is the activity labeled "Select best task". This could be as simple as selecting the oldest task that is eligible for assignment, or it could be as sophisticated as a stochastic optimization of expected throughput. The point is that the AASM approach carefully isolates this decision logic.

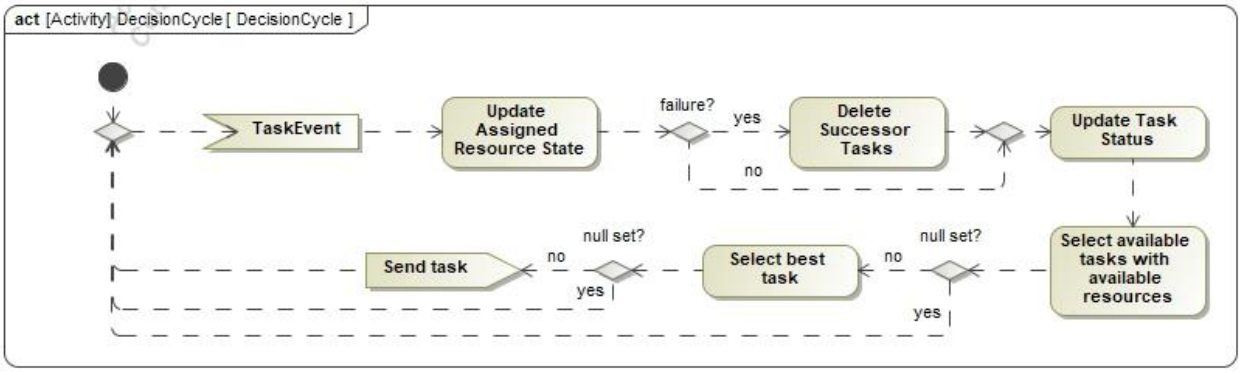


Figure 5: Controller Decision Cycle.

Finally, it is critical to note that the AASM approach to modeling control involves modeling the information that must be available to the controller for decision-making. Far too often, this kind of data is viewed as "not available" in standard simulation software applications, leading to significant simplification of the control model. As we move into the era of "smart factories" this level of simplification may no longer be acceptable.

4.5 Modeling Staff System

The active resources of the *StaffSystem* are *PO1*, *PO2* and *MT*. As shown in Figure 3(d), every *PO* has a set of capabilities, or operations that can be performed—setup, load, unload—and a set of qualifications. *PO1* is a specialization of *PO* that is qualified on diffusion and implant, while *PO2* is a specialization that is qualified on implant and litho. The model for *MT* is similar. Both *PO* and *MT* are specializations of *Staff*, with capabilities for meetings, meals and breaks.

Our AASM for IMF must allow us to alter the number and perhaps qualifications of the operators and techs, which means that cells must request specific services, but not specific resources. This is a general principle for modeling DELS, i.e., what is exposed is the capability, but the detailed implementation is hidden.

The *StaffSystem* will receive requests from cells for operations (setup, load, unload) and maintenance (PM, EM) and will decide when services are provided and by what staff. In addition, the *StaffSystem* must create tasks corresponding to individual staff requirements for breaks and meetings. These tasks can have due dates to insure they are completed within the staff member's shift. Figure 6 illustrates the states and state transitions for the PO. The MT has a similar set of states and transitions.

Control of the *StaffSystem* is very similar to control of the *IMF*. The *StaffSystemController* maintains a list of tasks, and each task is in one of four states—pending, available, assigned, or completed. Whenever a staff completes a task, the *StaffSystemController* will respond in a manner very similar to Figure 5, updating staff and task state, determining the set of tasks that can be assigned to staff, and choosing the best task to assign next. One additional step will be notifying the requesting cell that the task has been completed. Of course, there are a number of considerations in making the assignment decision, such as continuity of operation; if a PO unloads a machine, it would make sense to not leave the machine idle if there are lots waiting in the cells stocker that could be loaded. It also is worth noting that the model could be elaborated in interesting ways. For example, when a cell requests a load operation, it might at the same time request an unload operation with a start time equal to the completion time of the load operation plus the expected run time of the step.

Note that some tasks are resource-type specific, e.g., load or unload, whereas some task are resource-specific, e.g., *PO1* meeting task. Also, if the staff is not currently at the location corresponding to a task assigned then the staff must "go" to the correct location, i.e., the *StaffSystemController* must authorize the relocation and it must be completed prior to beginning the assigned task.

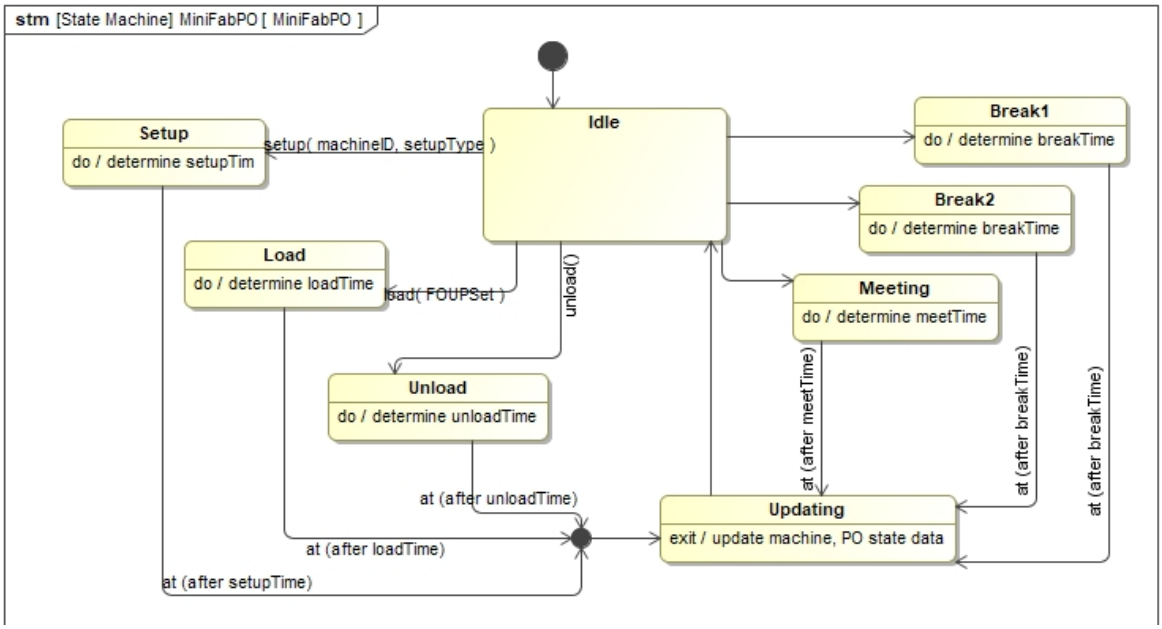


Figure 6: Production Operator State Model.

Note also the interesting interaction and synchronization among the cell controller, the *StaffSystemController*, the PO and the cell stocker. The simplest implementation for the load operation would be for the PO to execute a direct call to the stocker's put operation, rather than requiring communication from the PO to the *StaffSystemController* to the cell controller to the stocker.

4.6 Cells

Figure 3(c) illustrates the cell model for *Cell1*. Each cell in IMF has six "products", corresponding to each of the three products in the *ProductSet* after one of two specific process steps. For example, *Cell1* produces the two diffusion steps for each of the three products. The exposed capabilities of the cells are these six process steps, which appear in the IMF capability models illustrated in Figure 3(c).

The cell resources are the stocker and the process tools. As discussed earlier, the relevant stocker capabilities are retrieve and putaway from the PO interface, i.e., retrieve to a PO and putaway from a PO.

For each of the six process steps, the cell process involves having a PO retrieve a batch (perhaps of size 1) of FOUPs from the cell's stocker, do setup if required, load them into a machine, run the required process step on the machine, unload the FOUPs and return them to the stocker. Note the PO does not have to be present while the process step is run. In addition, for each machine there is a PM process model and corresponding requests for PM service from the *StaffSystem*.

In the formulation of the AASM presented here, the cell controller will request service from the *StaffSystem* both for production operations and for maintenance. When a FOUP is returned to the cell stocker, the cell controller will report to the *IMFController* that the corresponding cell operation is complete. The control decision cycle for the cell will be very similar to that shown in Figure 5. The key decisions the cell controller must make correspond to the selection of batches, and the sequencing of requests to the *StaffSystem*. These decisions must conform to the constraints specified for batches in cell 1, and the alternation of machines for *TW* in cells 1 and 2.

Clearly, alternate formulations are possible, e.g., having the staff temporarily assigned to the cells and taking tasks directly from the cell controller. A bit of reflection on this will reveal that it is more difficult to model, to implement, and to control, because a cell controller will not have access to the information about tasks that are pending in other cells.

5 CONCLUDING COMMENTS

Simulation is never an end unto itself; rather it is a means to support decision-making, usually about the design of some aspect of the system of interest. The value of simulation is always going to be limited by its fidelity, or lack of it, in representing that system of interest. What this paper attempts to demonstrate is the potential for simulation-agnostic system models to help bridge from the system of interest to a simulation of the system of interest.

Kempf's simple system, the *IMF* (Kempf 1994) has been modeled using OMG SysML ™, and most of the key elements have been presented and discussed. Many more details are incorporated in the full SysML model, available from the author. What should be clear from the presentation, however, is that this modelling approach provides a comprehensive "requirements document" for the development of a simulation model, structured in a way that would support relatively easy experimentation with alternative system specifications, especially for operational controls.

A few generic modeling principles have been illustrated. An object-oriented approach was used, in which active resources expose their capabilities but not their inner working details of how capabilities are realized. This approach enables "control encapsulation" which is one way to organize control architecture. A publish/subscribe messaging paradigm is a functional representation of communication within the system of interest. Although actual implementation may be quite different, it still must provide the same functionality.

The model presented here is a "first generation" AASM and there are a number of opportunities to improve it. For example, resource capabilities are modeled as both a *capabilitySet* listing the processes, and as operations that can be called. This works, but seems clumsy, and perhaps a more elegant modeling approach can be developed. Existing data models, such as (Laipple *et al* 2018) should be integrated.

There are a number of modeling challenges not addressed here but requiring resolution in more ambitious system models. A formal approach is needed for modeling batching and setup constraints, so they can be treated as input specifications, rather than hard-coded in analysis models. There are "on the

fly" routing decisions that aren't addressed in the IMF model; one example is routing FOUPs to single lot stockers, which requires deciding the destination. A good and generic approach is needed for modeling the relationship between a product's processing requirements, as illustrated in Figure 4(b) and the capabilities defined for resources. The model presented here does not use a standard model for time; calendars and schedules need definition, as does the integration of calendars and schedules with resource behaviors. Not addressed is the necessary hand-off of tasks between staff when a process extends over a shift boundary. These and other developments do not require new fundamental research; rather they require adapting existing modeling approaches to the formalisms of SysML.

It should be clear that creating the IMF AASM was not a trivial undertaking. Many person-hours were expended to develop and revise the model. A reasonable question is "Why do it?" There are both tactical and strategic reasons. From a tactical perspective, creating the SysML model requires making explicit every assumption about the IMF—what are its parts and how they are connected, what decisions must be made, how they are made and how decisions are made actionable. The resulting SysML model not only provides a platform on which stakeholders can understand and approve assumptions about the system of interest, it also provides a platform on which the simulation analyst can construct a high-fidelity model of the system.

From a strategic perspective, the development of AASMs based on standard representations creates the opportunity to automate a great deal of the simulation model development process. Simulation components corresponding to resources and controllers can be configured and connected using the information contained in the SysML model (see (Batarseh *et al* 2016) (Sprock and McGinnis 2016) and (Thiers *et al* 2016) for examples of this capability). If the stakeholders have approved the AASM (which they can understand), the conforming simulation model acquires significant "face validity".

Finally, the model presented here explicitly addresses the operations management decisions required to manage the flow of lots through the *IMF*. Operational control represents the "final frontier" for simulation analysis of production systems, and a challenge that absolutely must be met if the promise of smart manufacturing and digital factories is to be fulfilled. The technology to support AASM for DELS is one way to attack that challenge.

REFERENCES

Batarseh, O. G., E. Huang, and L. McGinnis. 2016. "Capturing simulation tool and application domain knowledge for automating simulation model creation," *Journal of Simulation*, Vol. 9, Issue 1, pp1-15.

Kempf, K. 1994. "Intel Five-Machine Six Step Mini-Fab Description," https://aar.faculty.asu.edu/research/intel/papers/fabspec.html, accessed April 12, 2020.

Laipple, G., S. Dauzere-Peres, T. Ponsignon, and P. Vialletelle. 2018. "Generic Data Model for Semiconductor Manufacturing Supply Chains". In *Proceedings of the 2018 Winter Simulation Conference*, edited by M. Rabe, A.A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, 3615–3626. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Lin, P. and L. F. McGinnis.2017. "Test Problems, Reference Models and Fab Simulation". In *Proceedings of the 2017 Winter Simulation Conference*, edited by W.K.V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, and E. Page, 3624-3635. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

McGinnis, L. 2019. Formalizing *ISA-95 Level 3 Control with Smart Manufacturing System Models*. https://doi.org/10.6028/NIST.GCR.19-022, accessed April 10, 2020.

McGinnis, L. F., E. Huang, and K. Wu. 2006. "Systems Engineering and Design of High-Tech Factories." In *Proceedings of the 2006 Winter Simulation Conference*, edited by L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, 1880-1886. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Nazzal, D. and L. F. McGinnis. 2006. "An Analytical Model of Vehicle-Based Automated Material Handling Systems in Semiconductor Fabs". In *Proceedings of the 2006 Winter Simulation Conference*, edited by L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, 1871-1879. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Sprock, T. 2016. *A metamodel of operational control for discrete event logistics systems*. PhD Thesis, School of Industrial and Systems Engineering, Georgia Institute of Technology. http://hdl.handle.net/1853/54946, accessed April 10, 2020.

Sprock, T., and L. F. McGinnis. 2016. "Simulation Optimization in Discrete Event Logistics Systems: The Challenge of Operational Control". In *Proceedings of the 2016 Winter Simulation Conference*, edited by T. M. K. Roeder, P. I. Frazier, R. Szechtman,

E. Zhou, T. Huschka, and S. E. Chick, 1170-1181, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

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

Sprock, T., G. Thiers, L. McGinnis, and C. Bock, C. 2019. *Theory of Discrete Event Logistics Systems (DELS) Specification*. NIST Interagency/Internal Report. https://www.overleaf.com/project/5bca3f4898fbf165b132f596, accessed April 10, 2020.

Thiers, G. 2015. *A model-based systems engineering methodology to make engineering analysis of discrete-event logistics systems more cost-accessible*. PhD Thesis, School of Industrial and Systems Engineering, Georgia Institute of Technology. http://hdl.handle.net/1853/52259, accessed April 10, 2020.

Thiers, G., T. Sprock, L. McGinnis, A. Graunke, and M. Christian. 2016. "Automated Production System Simulations Using Commercial Off-the-Shelf Simulation Tools". In *Proceedings of the 2016 Winter Simulation Conference*, edited by T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S. E. Chick, 1036–1047. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

AUTHOR BIOGRAPHIES

LEON F. McGINNIS Professor Emeritus in the Stewart School of Industrial and Systems Engineering at The Georgia Institute of Technology, where he has held leadership positions in faculty governance, academic programs and research centers. His research is focused on developing the methods and tools necessary for systems engineering of discrete event logistics systems, including design, planning, management and control issues. His email address is leon.mcginnis@gatech.edu and his work is featured on the website http://factory.isye.gatech.edu.