THE CHALLENGE OF PLANT-CONTROL SEPARATION IN WAFER FAB SIMULATIONS

LEON MCGINNIS
Goal: develop an implementable architecture for ISA 95 Level 3 smart manufacturing operations manager (SMOM). Consider interactions to Levels 2, 1, 0 for process control and Level 4 for planning. Demonstrate the architecture through testbed implementations representing two distinct application environments—central fill pharmacy and semiconductor wafer fab.
Will simulation even matter in the new world of AI, Big Data, Machine Cognition, Machine Learning, IIoT, Industrie 4.0, .... ?
If it does, then we MUST do something to make factory simulation better, faster and cheaper, because today it is not accurate enough, takes far too long to develop and use and costs far too much in terms of limited human resources.
HOW HAVE WE (RESEARCHERS) THOUGHT ABOUT MANUFACTURING?

Conceptual paradigm underlying all commercial simulation languages
The only “control” possible is the selection from a queue

We call it a “network of queues”, not a “network of servers”
WHAT IS MANUFACTURING REALLY?

The fab is:

- Network of resources—OHT, stockers, process tools
- Each with specific behaviors, or capabilities to execute processes
- Product—foups—move through this network of resources, where resources execute processes to transform the product to a more valuable state
- Control systems tell these resources which behaviors to execute and when
- An example of a ‘discrete event logistics systems”, or DELS
FUNDAMENTAL DISCONNECTS

• The queueing paradigm has a very limited capability for representing control
  • This is a crippling limitation for researchers
  • This is a tremendous cost sink for practitioners

• In manufacturing, there are no queues, only resources with behavioral capabilities
  (this is not quite true—I’ll discuss this more in a minute)
  • Control engineers write software that drives these resources

• Up until now, there has not been an effective methodology for bridging the gulf
  between the “language of queues” and the “language of control”
BRIDGING THE GULF

- Formal models of discrete event logistics systems, capturing resources, interfaces, behaviors, controls, products, processes
- At multiple levels of abstraction (just like for integrated circuits!)
- Ability to use models of DELS instances as the baseline for creating decision support analysis models (so our results can be effectively communicated to controls engineers and actually used!)
  - Even if the decision support is an agent/machine intelligence/AI....

A suggested approach to bridging the gulf
GOAL: A USEFUL “VIRTUAL FACTORY”
GOAL: A USEFUL “VIRTUAL FACTORY”
GOAL: A USEFUL “VIRTUAL FACTORY”

Adjustable Synthetic Data

Order Stream Specification

Control System Specification

Base System Specification

Component Libraries

Simulation Model Generator

TH

CT

WIP
Re-entrant flow
Setup, batch, machine failure
Preventive and breakdown repair
Operators with breaks and mtgs
Transporter and stockers
If we are going to separate plant and control, we must have some conceptual model of “controller”
Controller requirements
- Event driven decision-making
- State-based decision-making
- All actuation via base system
- Decision support has well-defined interface
DEFINING DELS

[Diagram showing the relationships between DELS, Base System, Controller,稳步, Monitor, Plant Model, Decision Support, Task Definer, and their attributes and methods.]

- DELS
  - ownedBaseSystem
  - ownedController
  - usedMsgHandler
  - values
    - name: String

- Base System
  - mover
  - store

- Controller
  - monitor
  - plantModel
  - decider

-稳步
  - Task Definer
  - Maker
  - Mover

- Monitor
  - Plant Model

- Decision Support
  - Task Definer

- Task Definer
  - Task Definer
For a process to be executable, there must be some resource in the base system having the capability to execute that process (or the capability to be reconfigured to execute that process, where the reconfiguration itself is a behavioral capability).

A product has an associate process, and that process can “nest” multiple ‘sub’ processes, with constraints such as precedence, timing, etc.
We must be able to precisely define what flows on the connections, and the detailed structure of the interfaces.

“Task” is used here to represent both “commands” to the base system and “event” messages from the base system.
DOMAIN SPECIFIC CUSTOMIZATION
FAB DOMAIN DELS MODEL

```
<<block>>
Base System

parts
maker : Maker [1..*]
mover : Mover [0..*]
store : Store [0..*]
```

```
<<block>>
Fab Bay

fabBay
1..*
```

```
<<block>>
FabBase System

fabOHT
1..*
```

```
<<block>>
OHT
```

```
<<block>>
Stocker

fabStock
1..*
```

```
<<block>>
Fab

^ownedController : Controller

ownedBaseSystem : FabBaseSystem (redefines ownedBaseSystem)

^usedMsgHandler : MsgHandler

^name : String
```

---

CREATING THE NEXT®
FAB LEVEL ABSTRACTION

Only one “bay” is shown, but more don’t change the fundamental structure
INTEL MINIFAB DOMAIN
PROCESS TOOL BEHAVIOR (10 STATES)
OHT BEHAVIOR
STOCKER BEHAVIOR

- **gettingFromOHT**: exit / update buffer state table
  - OHT calls Get operation
  - Get operation completes
  - Controller calls Put operation
  - Put operation completes
  - puttingToOHT: exit / update buffer state table

- **gettingFromPO**: exit / update buffer state table
  - PO completes unload
  - Get operation completes
  - PO starts load
  - puttingToPO: exit / update buffer state table
OPERATOR BEHAVIOR

- **Moving**
  - moveTo(cell())
  - at (after moveTime)
  - do / determine moveTime

- **Setup**
  - load()
  - at (after setupTime)
  - do / determine setupTime

- **Load**
  - do / determine loadTime
  - at (after loadTime)

- **UnLoad**
  - do / determine unloadTime
  - at (after unloadTime)
  - at (after meetTime)

- **Idle**
  - setup()

- **Meeting**
  - meet()
  - do / determine meetTime

- **Break1**
  - takeBreak()
  - do / determine breakTime

- **Break2**
  - takeBreak()
  - do / determine breakTime

- **Updating**
  - exit / update machine, PO state data
CELL CONTROLLER BEHAVIOR

[Diagram showing the behavior of a cell controller.]

1. **Idle**
   - [load complete]
   - [setup, unload, run, PM or EM complete]

2. **StartingMachine**
   - do / start loaded machine process

3. **UpdatingBufferState**
   - do / update buffer table

4. **UpdatingMachineState**
   - do / update machine table

5. **UpdatingOperatorAvail**
   - do / update operator table

6. **IdentifyingCandidateTasks**
   - do / determine all executable tasks

7. **SelectingPreferredTask**
   - do / select best task to execute next

8. **AssigningTask**
   - do / assign task to operator
MINIFAB CONTROLLER BEHAVIOR
SO WHAT?

- This model captures all the (relevant) behaviors of the resources and the controllers.
- Control decisions are described in terms of the behavior that implements them.
- This model captures the events that trigger control behaviors.
- The base system state model must provide the information needed by the decision making function (decision support function).
SUPPOSE WE WANT TO SIMULATE

Control System Model

- Base System Model
- Monitor
- Decision Support

Base System Model

- Event Calendar
- Resource Model

“Other” Implementation

Simulation Infrastructure

- Autosched
- Automod
SUPPOSE WE WANT TO SIMULATE

We have demonstrated, in multiple domains, that it is feasible to autogenerate the simulation model of the base system. What is required is a set of simulation components that can be mapped to the reference model, and populated with instance data.

The remaining challenge is to create a control system reference model, with generic components having well-defined interfaces, so that particular decision support methods can be encapsulated. Then for a particular application, only the decision support code needs to be hand crafted. For standard applications, even that might become library models.
FUTURE WORK

- We are doing this with Mathworks SimEvents and MATLAB (SimEvents permits “MATLAB function blocks” which we use to implement the controller)

- Any DES should be able to provide the same access to an underlying programming language and data space

- For a specific domain, e.g., wafer fabs, can we identify a set of generic controller functions, and a generic schema for base systems state, so that libraries of DELS controller components can be distributed?

- If we can do this for analysis of operations management decisions, can we extend these ideas to the interface between operations management and production planning?

- Can we extend these ideas to the modeling and analysis of supply chains?
  - Hint: the answer is yes
WHY WOULD WE WANT TO?

Because, whether we are designing conventional control systems, or developing “intelligent agents”, we will always need a “laboratory” in which we can train/test our designs. Setting them loose in the real world without such testing or training is not a feasible option.