Proceedings of the 2003 Winter Simulation Conference
S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds.

SEMICONDUCTOR SUPPLY NETWORK SIMULATION

Gary W. Godding

Component Automation Systems
Intel Corporation
5000 W. Chandler Blvd – MS CH3-68
Chandler, AZ 85226, U.S.A.

Hessam S. Sarjoughian

Arizona Center for Integrative Modeling & Simulation
Computer Science & Engineering Dept.
Arizona State University
Tempe, AZ 85287-5406, U.S.A.

Karl G. Kempf

Decision Technologies
Intel Corporation
5000 W. Chandler Blvd – MS CH3-111
Chandler, AZ 85226, U.S.A.

ABSTRACT

More efficient and effective control of supply networks is conservatively worth billions of dollars to the national and world economy. Developing improved control requires simulation of physical flows of materials involved and decision policies governing these flows. This paper describes our initial work on modeling each of these flows as well as simulating their integration through the synchronized interchange of data. We show the level of abstraction that is appropriate, formulate and test a representative model, and describe our findings and conclusions.

1 PROBLEM DESCRIPTION

Improving the operational efficiency of supply networks for physical goods is one way to boost the national and global economies. Understanding the control physics of this kind of network is key to providing continuous improvement. Developing such an understanding requires extensive experimentation to formulate and validate control policies. Since this type of network generates enormous wealth, direct experimentation involves untenable financial risk. Some form of modeling and simulation is required.

Simulating supply networks requires the modeling of different but interrelated flows. Physical flows represent the goods being produced, stored, and shipped including raw materials, work in progress, and finished goods. Data flow represents the past, present, and forecast future state of the physical flows as well as the demands from the marketplace. Such data is used by policies controlling the balance between variable demand and variable supply to maximize efficiency. The resulting decision flows describe when and how much of which materials to release into manufacturing facilities and transportation links as well as how much inventory to hold in warehouses.

The goal is to provide customer service to maximize revenue (the right quantity of the right product in the right place at the right time) while minimizing costs (materials, production, storage, transport, obsolesce) to maximize profits now and in the future. This requires a number of difficult tradeoffs. For example, to maximize customer service, high volumes of finished product might be stored close to the customer. But for products with short effective lives, costs are minimized by holding raw materials at the beginning of manufacturing lines until firm orders are placed. Decision policies are required to manage such tradeoffs over time, and the quality measure of policies is supply network profitability.

We have developed a software architecture to model and simulate material, data, and decision flows and the interactions between them. Using this architecture a wide variety of experiments can be implemented to explore the behavior of supply networks that we demonstrate with a typical problem from semiconductor manufacturing. This test problem shows our approach to the granularity of the objects to be modeled and the granularity of time over which to simulate. We address the division of functionality between the physical module and the decision module as well as the data passing requirements including synchronization. We explore the ease of modifying the physical entities (in both parameters and topology) and the decision policy.

Although this initial demonstration covers multiple time periods, during a simulation the network topology and

products (to mention but a few) are fixed. We believe however, that the architecture will support the dynamics of actual semiconductor supply networks where new factories come on line and new products are introduced (and other dynamic changes) as a simulation unfolds over time. Furthermore, we are preparing to test the architecture over multiple dimensions of scalability. The first is size scalability in terms of number of factories, products, customers, warehouses, transport links, etc., included in the material flow. The second is complexity scalability as the relationships between product age and product pricing, or between multiple customers for the same scarce product, or other such complexities are included in the decision flow.

2 RELATED WORK

A number of methods have been explored to model and simulate supply networks. These include principally discrete event simulation, agent-based simulation, and systems dynamics simulation. Other less well explored approaches include fuzzy sets and spreadsheet analysis.

2.1 Discrete Event Simulation

For modeling and simulation of the physical flows in a supply network, discrete event simulation (DES) might be considered the most obvious choice since it has been used extensively in the study of material flow in manufacturing systems at the tool and factory level (Law and Kelton 1991) . It has been shown that using the appropriate abstraction, DES can be used for the factory components of a supply network (Godding and Kempf 2001; Kempf, Knutson, Fowler, Armbruster, Babu, Duarte 2001). However, DES does not lend itself as easily to decision flows. While there are DES systems that include the simple kinds of dispatching rules that can be used to control a set of tools, more complex policies generally require user exits and coding in some basic programming language. Rapid experimentation with control policies across complex supply networks is not within the grasp of DES.

2.2 Agent-Based Modeling

Agent-based modeling is intended to support a variety of behaviors that depend on autonomy, mobility, and rationality where handling independent and cooperative decision-making is necessary. However, unlike some other modeling approaches (e.g., discrete-event), to date there does not exist any universally accepted theory for modeling agent dynamics. Agent-based modeling, nonetheless, plays a principal role in describing behavior that does not lend itself to a priori reasoning. In particular, in domains such as enterprise engineering, decision-making is inherently non-monotonic. Agent-based modeling can model important autonomy and mobility traits (e.g., Muller and Pischel 1994) to deal with unpredictable conditions often present in supply network and logistics. Indeed, agent-based modeling is widely accepted to be necessary where local and cooperative planning is required. However, given the level of complexity required for this kind of agent-based modeling, it should not be used if simpler modeling techniques can be used. For example, other researchers (e.g., Swaminathan, Smith, and Sadeh 1998) apply simpler types of agents, called software components, to model static and dynamic aspects of supply network entities such as transportation agents.

2.3 System Dynamics

There has been much recent research using Systems Dynamics (SD) for supply network modeling (Angerhofer and Angelides 2000). The use of SD, formally known as industrial dynamics, was first suggested in 1958 (Forrester 1958). The SD approach for analyzing behavior of complex systems includes (1) modeling the system as feedback loops and delays, (2) turning the model into a set of quantitative equations, and (3) using simulation to observe the behavior of the system. Users follow this concept of modeling and simulating system behavior to gain insights on control policies. However, the modeling and numerical simulation methods provided by current SD environments do not provide the granularity needed to model the complex stochastic material flows and associated control algorithms for a semiconductor supply network without significant extension.

3 APPROACH

In developing a semiconductor supply network simulation model, it is useful to distinguish between material processing (low-level) and decision-making (high-level) dynamics. Abstracting a complex system into low-level and high-level layers supports using distinct modeling approaches as appropriate (Sarjoughian, Zeigler, Hall 2001). For example, characterization of physical flow of materials independently from the modeling of decision-making offers a basis for synthesizing a supply network using different modeling methods (e.g., discrete event for materials and mathematical optimization for decisions).

To illustrate the significance of this separation, consider Figure 1 where the decision and physical processing layers are defined. The decision layer is responsible for making higher-level decisions (controls) based on the lower-level behavior (data) observed from the physical processing layer. At the decision layer, multiple components can collaborate to generate control commands for the physical processing layer. The physical processing layer can represent processes, inventory, and shipping components as well as their interactions. In this layer, the products generated in the process are stored in the inventory be-

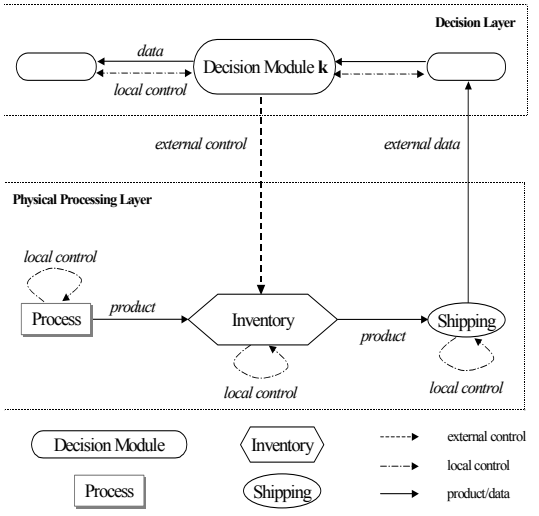

Figure 1: Decision and Physical Processing Layers and their Interactions.

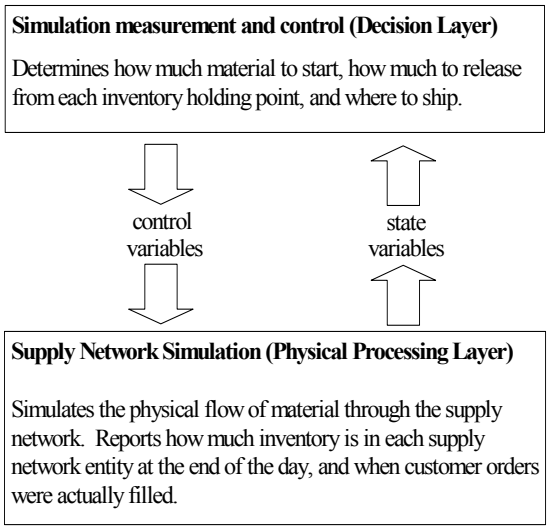

Figure 2: Two Layer Abstract View of Supply Network

fore being shipped. Components from these two layers can interact with one another using external control and data as shown in Figure 1.

The interactions between the decision and physical processing layers are governed by the former sending external control commands to the latter. Examples of the external control for the physical processing line are "how much to release into the line during some time interval", "how to configure material to demand," and "which warehouse the material should be shipped to". Correspondingly the decision layer receives the state of material processing for determining external control commands. Examples of data sent by the lower-level (and consumed by the higher-level) are "the amount of semi-finished goods available for producing some desired finished goods" and "the amount of product shipped to customer and Geo-warehouse". The separation of material processes from decision-making is based on distinguishing data, product, local and external controls from one another. The external control and data between these two layers allow isolating tactical (local control) and strategic decision-making (external control) from one another. Figure 2 depicts an abstract view of the physical processing and decision layers, their interactions, and their collective role in fulfilling end-to-end demands of a supply network.

To further illustrate the concept of separation of concerns as shown in Figures 1 and 2, consider a segment of a real-world distribution network which has an assembly test factory and a logistics network. The decision to separate decision-making (inventory holding policy) from material processing plays a vital role in model development of a supply network. For example, the decision-making algorithms tend to consider cost/revenue tradeoffs by controlling how much material to build and where to store it while the physical model considers physical characteristics and constraints of how material flows through the network (e.g. capacity, delay, and yield).

The physical processing layer of a semiconductor supply network has inventory holdings, processes, shipping, and customer entities (see Figure 3). In this setting, material arriving is stored at the die inventory and processed into different types of products according to physical characteristics of the die and some types of local and external decision policy. Each of the die flows through the assembly test line, is assembled into packages, split into categories based on test results, and stored into unfinished inventory. When the material is released from the unfinished inventory point, it is configured to a specific product which is ready for shipment to customers. The processing steps in the foregoing description can be categorized into material processing (e.g., categorizing die based on test results)

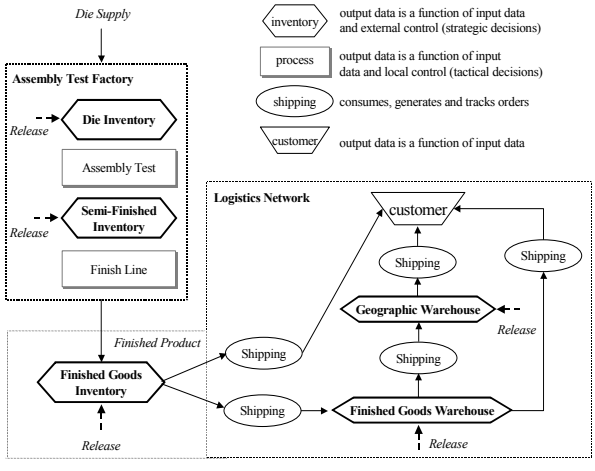

Figure 3: Model for Physical Processing Layer of a Semiconductor Supply Network

and decision-making (e.g., release a certain number of die to a given customer).

3.1 Modeling and Simulation Methodology

As stated earlier, having the ability to use different modeling techniques for the decision and physical manufacturing layer offers important advantages compared with monolithic methodologies where one modeling approach is used for describing all types of dynamics (e.g., high-level decision-making and low-level physical processes). For example, while discrete event modeling is well suited for manufacturing processes, it may be necessary and advantageous to use other techniques such as logic-based reasoning to model intelligent decision-making. For modeling the physical layer dynamics, we use DEVSJAVA software environment (ACIMS 2003) and its accompanying DEVS (Discrete Event System Specification) modeling and simulation methodology (Zeigler, Praehofer, and Kim 2000) (see Figure 4). For simplicity, in this study high-level decision algorithms are wrapped inside atomic DEVS model components. This choice allows executing our existing model using alternative simulation engines (e.g., parallel execution) in a straightforward manner.

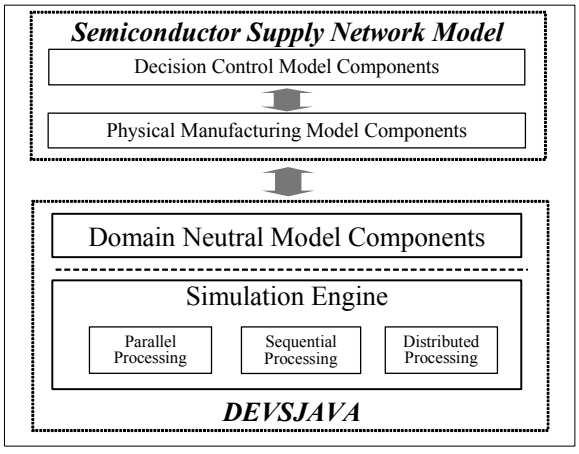

Figure 4: Supply Network Modeling and Simulation Environment

4 MODEL DEVELOPMENT

In DEVSJAVA, a model is either atomic or coupled. The former specifies inputs, outputs, states, state and output transitions, and timing function. The latter specifies hierarchical composite models where its components can be atomic or coupled models. Each coupled model can have inputs, outputs, and couplings among its children.

The supply network topology shown in Figure 3 has been modeled using the DEVSJAVA framework. First, the physical and decision layers were identified and decomposed into DEVSJAVA models. Second, the message objects were defined. Finally, the message flow and synchronization were defined and modeled.

4.1 Physical Model Decomposition

The DEVSJAVA environment supports inheritance for creating atomic models and this feature has been used extensively in creating our model. The class hierarchy is shown in Figure 5. The *Viewable Atomic* model is a DEVSJAVA base model from which all the other atomic models are derived. The *Supply Network Entity* model provides common methods for synchronization, inventory reporting, capacity management, and delay.

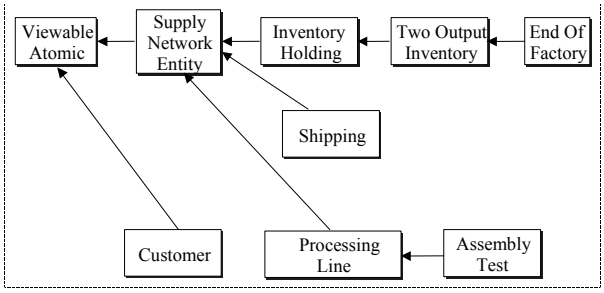

Figure 5: Object Hierarchy – Supply Network Components and Customer.

The physical model has been decomposed into four major categories of entities including processing lines, inventory holdings, shipping, and customers. Specializations have been created to support additional functionality. Descriptions of each follow, and the detailed model is shown in Figure 6.

4.1.1 Inventory Holding

The inventory holding is modeled to behave like a store or a warehouse. When material arrives, it is added to the current pool of inventory. Material will not leave until an explicit release signal is received. The inventory holding is the only entity in the physical model that receives commands. Control of the physical network is accomplished by controlling how much, what, and where to release material from the inventory holding points. Inventory holding points can have capacity and delay.

The basic inventory holding has one input and one output for material flow. There have been two specializations created; the two output inventory, and the end-of-factory inventory. The two output inventory has different paths through which material can leave. The end-of-factory model is a specialization of the two output model. End-of-factory model is used to simulate the back of a factory where there is limited storage space. If any material remains in an end-of-factory model at the end of a time period, the material is automatically released to the default output path.

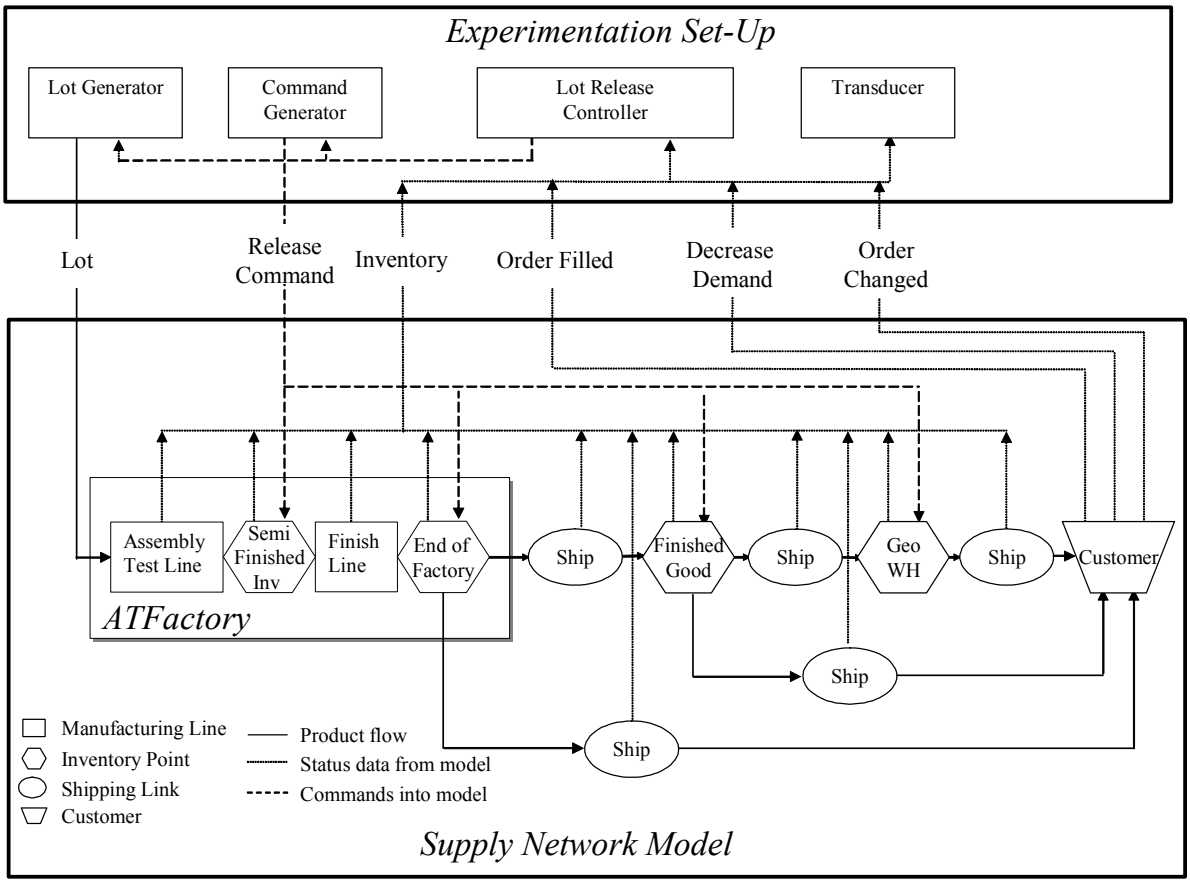

Figure 6: Detailed Model of the Supply Network

4.1.2 Shipping

Material is delayed to simulate the transport time. Shipping entities have capacity and delay. Shipping by different methods (air/land/sea) could be modeled and would have different cost/time tradeoffs to explore.

4.1.3 Processing Line

Processing lines model manufacturing links or assembly operations. Processing lines can change the material. These lines have capacity, delay, and yield. A simple example is the finish line used in our supply network model. A specialization of the processing line has been created to simulate the assembly test operation. The assembly test operation splits one input product into three output products based on a stochastic distribution. This distribution simulates the behavior of a semiconductor test operation where product is divided based on physical characteristics of the die.

4.1.4 Customer

The customer can consume orders during each time period. Customers may change orders within a certain time window from the order due date and they may cancel late orders. If customer cancels more than two orders, average orders from that customer could decrease. This is intended to simulate the behavior of customers switching to an alternate supplier when service level falls.

4.1.5 ATFactory

The ATFactory (AssemblyTest factory) is a coupled model that consists of an assembly test line, a semi finished inventory holding, a finish line, and an end-of-factory inventory holding.

4.2 Decision Model Decomposition

4.2.1 Decision Layer Atomic Models

The decision model is made up of four different atomic models including a lot release controller, a lot generator, a command generator, and a transducer (see Figure 6). The lot release controller implements the heuristic described in the next section. The lot generator inputs new material into the simulation, the command generator inputs commands. The transducer collects data from the simulation for off-line analysis.

4.2.2 Decision Algorithm

A heuristic has been implemented for the control algorithm. The heuristic attempts to keep inventory levels at a target level, while meeting customer orders on time. The heuristic can only consider a single shipping path, i.e. product is shipped to customer via the geo warehouse, geo warehouse is replenished from finished goods warehouse, and finished goods is replenished from the ATFactory.

The heuristic calculates the difference between expected pipeline inventory and the actual inventory. If the difference is positive, it will release that much for the day. The heuristic is used to calculate both how many lots to start and how much material to release from the warehouses. The required inputs to the heuristic are: total inventory currently in the supply network pipeline, the orders that have been filled, the forecast demand over the time period of the network delay, the average delay of the pipeline, and the inventory targets. Additional details of the heuristic can be found in (Armbruster, Chidambaram, Godding, and Kempf 2001). The inventory and order data is obtained from the physical network during the runtime. The other data is input as parameters to the model.

4.3 Message Types

Three types of messages have been defined to flow between the different components; the lot, an order, and a command. These messages have been derived from the DEVSJAVA entity class.

The hierarchy is shown in Figure 7. The lot is the primary unit of material that flows through the network. It contains the quantities of the three different products, and has some state variables used to track how long the material has been processed within the different entities. The command is an extension of the lot and has additional fields for specifying output path and special instructions. The command is sent to the inventory holding points to tell how much material to release, which output path to release on, and any special instruction for next entity such as which product the material should be configured to or which customer order should be filled. Orders are sent from the customer. Additional fields in orders are the order ID field, a due date, and a customer ID.

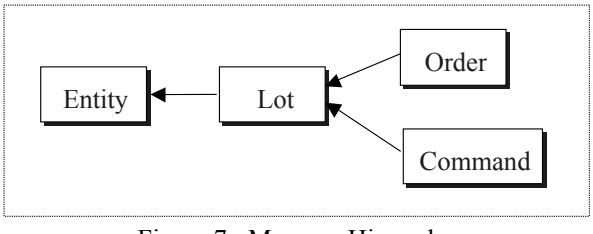

Figure 7: Message Hierarchy

4.4 Data Flow and Synchronization

Three types of flows have been identified in the model. There is material flow, control flow, and information flow (see Figure 6). Material flow represents the product being processed and consumed by the customers. Information flow is generated by the physical model and represents the externally known state of the physical network at any given time period. The control flow is generated by the decision model and sent to the physical model to control material flow. Material flow is controlled by how inventory is released.

4.4.1 Material Flow in Physical Model

Material flow starts from the lot generator in the decision layer. When the material is released into the assembly test line, it is split into three different products using a stochastic distribution. Processing time for assembly test is approximately 2 weeks, that is also drawn from a distribution. After material leaves assembly test, it flows into the semi finished inventory. Material has to be released from this inventory into the finish line. The release command sent to semi finished inventory contains an additional field specifying what product the material should be configured to. After material leaves the finished line, it flows into the end of factory inventory point. It can be shipped either to the customer or to finished goods from here. If it is shipped to a customer, an order ID has to be supplied to specify which customer order is being filled. After finished goods, materials can be shipped to either the customer or the geo warehouse.

The physical layer sends the following messages to the decision layer:

- Inventory level at the end of the day.
- Customer orders filled at the end of the day.
- Customer orders that have been changed.
- Customer demand decreased.

4.4.2 Decision Flow

At every time period the lot release controller module will look at the data received from the previous period to calculate the release for the next period. The lot release controller sends its output to the command and lot generator. The lot generator will send new material to the assembly test line and the command generator will send release commands to the inventory points.

The messages sent from the decision layer are:

- New material to release into the simulation.
- How much material to release from each inventory holding point.
- How to configure material in the finish line.
- Which orders should be filled with the available finished product.

4.4.3 Synchronization

The control for the model is based on knowledge of overall inventory at each entity and what orders have been filled at the end of a given time period. The decision algorithm must have this knowledge to make decisions for the next day. A clock is used to synchronize all entities by broadcasting an end of day message. The sequence of execution for the simulation, repeated until the simulation is complete, is given below.

1. Simulation starts a new time period. Processing starts on the new material and commands sent by the decision layer in the previous period. Processing continues on material already in the simulation.
2. Decision layer calculates amount of material to release into the simulation for the next time period. Calculation is completed by the middle of the current time period.
3. Decision layer calculates material to release from each warehouse for the next time period. Calculation is complete by three quarters of the current time period.
4. Decision Layer sends material and commands to the simulation at three quarters of current time period. (Processing of new material and commands will not start until the beginning of the next period).
5. The simulation completes the current time period. It reports to the decision layer how much inventory is at each entity and which orders were filled.

The clock module sends out a synchronization message at the end of every time period. The synchronization message is sent to all entities to report the timestamp of the next time period. All entities use this to timestamp messages and increment their internal time.

The transducer records all data into a data structure. At the end of the simulation, the transducer writes all data into CSV files to enable offline analysis.

5 SIMULATION RESULTS AND FINDINGS

A set of 12 experiments was run to validate the system behavior. Four types of demand signals were used with 3 different inventory targets. Simulation data was collected for 200 days. Experiment descriptions and results are shown in Table 1. For each type of demand, inventory targets of 500, 1000, and 2000 were tried. These targets were applied to all inventory points. Experiments 1 through 3 had a constant demand of 500 for all three products. Experiments 4 through 6 had a square wave demand that changed between 500 and 750 every 25 days for each product. Experiments 7 through 8 had a demand that increased by 100 every 25 days for each product, and experiments 10 through 12 had a demand selected from a uniform distribution between 0 and 1000. The measured results were the percent of orders received late by the customer, and the average number of days an order was late.

Table 1: Experimental Results

Exp #	Demand Profile	Inventory Target	Percent Orders Late	Ave. Days late
1	Constant 500	500	100%	3.405
2	Constant 500	1000	88%	1.03
3	Constant 500	2000	0	0
4	Square 500-750	500	100%	3.905
5	Square 500-750	1000	98%	1.74
6	Square 500-750	2000	1%	1
7	Increasing	500	100%	4.72
8	Increasing	1000	100%	3.09
9	Increasing	2000	55%	1.21
10	Random 0-1000	500	86%	3.71
11	Random 0-1000	1000	80%	2.23
12	Random 0-1000	2000	40%	1.08

It can be seen that the control had better results for steady demand (Experiment 1-6) than for increasing or random demands. The results also show that inventory played important role in improving customer service levels.

Data collected from experiment 6 is shown in Figures 8, 9, and 10. Figure 8 shows how much material was released into assembly test on each day. This quantity is close to the sum of demand for all three product because assembly test will split its input into 3 output products.

Figure 9 shows inventory levels of each product in the unfinished inventory store. The assembly test lot split distribution was %33 plus/minus %2 for products 1 and 3. Product 2 split was the remaining difference from product 1 and 3 results. (e.g. If values drawn for product 1 and 3 were %31 and %34, then product 2 split would be %35). The results show that product 2 inventory is climbing, which is expected since the split percentage should be slightly higher.

Inventory levels for finished goods is shown in Figure 10. Levels for all three products are centered around the tar-

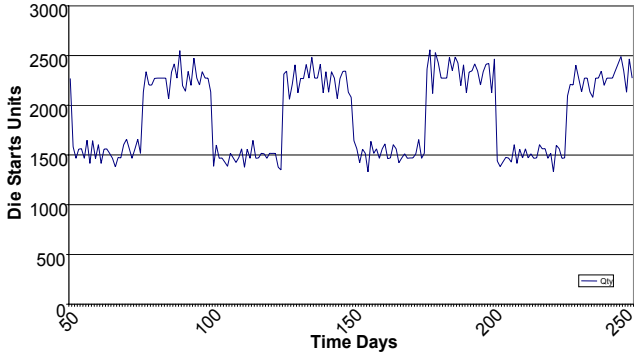

Figure 8: Die Released into Assembly/Test for Experiment 6

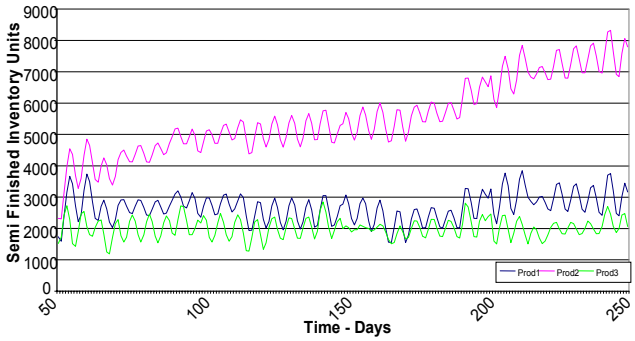

Figure 9: Unfinished Inventory Levels for Experiment 6

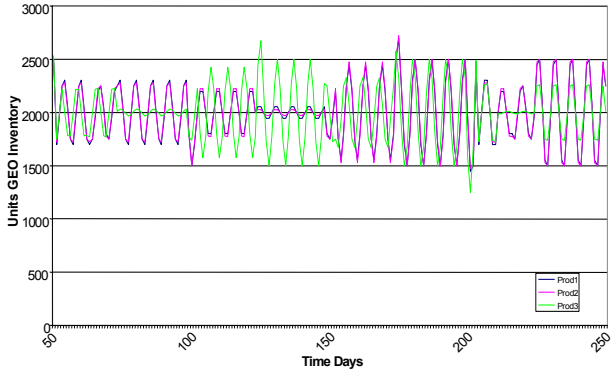

Figure 10: Finished Good Inventory for Experiment 6

get level at 2000. Oscillations are due to the control heuristic trying to compensate for the variability in assembly test.

Key findings of this effort include:

- The control algorithm quickly becomes complex for this small subset of a semiconductor supply network. With the control separated from the physical flow, it would be straightforward to try other types of control strategies. The supply network performance could be improved if the simple heuristic used in the initial testing were replaced with some more sophisticated control (e.g. mathematical optimization).
- Separating the physical flow from the decision engine simplifies the development effort for both the physical model and decision model. The interactions between control flow and physical flow is simplified.
- DEVSJAVA enabled modeling at a low enough level of granularity for both entity objects and simulation timing.
- The DEVSJAVA environment tied with this modeling approach provided good performance compared to previous work. A 300 day simulation took less than 2 minutes to run on a 500MHz Pentium III laptop with 128 MB of memory. A similar previous model in (Godding and Kempf 2001) had runtimes greater than 1 hour when run on a dual processor 900MHz server with 1GB of memory. This improved performance is a result of the more efficient modeling allowed by the DEVSJAVA environment, and perhaps from DEVSJAVA itself.

6 FOLLOW-ON WORK

6.1 Supply Network Model

This simulation study only included a subset of the semiconductor supply network. For the next phase, we will include multiple assembly test factories, multiple customers in different geographies, wafer fabrication plants, and add a complicated bill of materials. With some specialization, the atomic models implemented will support all these new capabilities.

The new structures will add many new complexities to the control algorithm. The addition of fabrication plants will multiply the cycle times. Multiple manufacturing facilities and customers will add interplant routing and logistic decisions, and the bill of materials will add additional routing decisions on how to build product.

6.2 Control Interface

Interface specifications will be formalized to enable the connection of different types of control applications. Much research is being done on control and it will be useful to see how the different approaches work (e.g. mathematical optimization, AI approaches, etc…). Issues such as what type of data to pass, how to transform the data, and the type of architecture required must be considered.

6.3 Multiple Modeling Formalisms

The approach described in Section 3 (see Figure 1) suggests the need for using distinct modeling techniques for the physical processing and decision layers. In this paper, however, we have separated the models according to the separation of concerns while using only one modeling formalism. Our ongoing research includes selecting a suitable modeling formalism for the decision layer and its integration with the DEVS formalism. Within such a framework, we will be able to formally devise an interface specification and demonstrate the impact and value of the proposed layering approach.

7 CONCLUSIONS

We have shown that the planning control algorithms can be separated out from the simulation of the actual physical flow of a small segment of the semiconductor supply network. Separation of these functions into two separate layers has not only simplified building the physical simulation, it has also provided an environment that facilitates experimentation with different types of control policies.

Both capabilities are very important, as scalability issues quickly become a problem when building realistic models for analyzing the supply network.

ACKNOWLEDGMENTS

This research is partially supported under the NSF Scaleable Enterprise Systems Grant No. DMI-0122227.

REFERENCES

ACIMS, DEVSJAVA Software, 2003. <http://www.acims.arizona.edu/SOFTWARE>.

Angerhofer, B. J. and M. C., Angelides, 2000. System Dynamic Modeling in Supply Chain Management: A Research Review, in *Proc. Winter Simulation Conference*, 342-351.

Armbruster, D., R. Chidambaram, G. W. Godding, K. G. Kempf, and I. Katzorke, 2001. Modeling and analysis of decision flows in complex supply networks, in *Proc. IV SIMPOI/POMS*, Sao Paulo, 1106-1114

Forrester, J.W. 1958. Industrial Dynamics: A Major Breakthrough for Decision Makers. *Harvard Business Review,* 36(4), 37-66.

Godding, G. W. and K. G. Kempf, 2001. A modular, scalable approach to modeling and analysis of semiconductor manufacturing supply chains, in *Proc. IV SIMPOI/POMS*, Sao Paulo, 1000-1007.

Kempf, K. G., K. Knutson, J. Fowler, D. Armbruster, P. Babu, and B. Duarte, 2001. Fast Accurate Simulation of Physical Flows in Demand Networks, in *Proc. Semiconductor Manufacturing Operational Modeling and Simulation Symposium*, Tempe, AZ, 111-116.

Law, A.M., and W. D. Kelton, 1991. *Simulation Design and Analysis*, McGraw-Hill Inc., New York.

Muller, J. P. and M. Pischel, 1994. An Architecture for Dynamically Interacting Agents, *International Journal of Intelligent and Cooperative Information Systems*, 3(1), 25-45.

Sarjoughian, H. S., B. P. Zeigler, S. B. Hall, 2001. A Layered Architecture for Agent-based System Development, *Proceedings of the IEEE*, 89(2), 201-213.

Swaminathan, J. M., S. F. Smith, N. M. Sadeh, 1998. Modeling Supply Chain Dynamics: A Multiagent Approach, *Decision Sciences*, 29(3), 607-632.

Zeigler, B. P., H. Praehofer, T. G. Kim, 2000. *Theory of Modeling and Simulation*, Second Edition, Academic Press.

AUTHOR BIOGRAPHIES

GARY W. GODDING is a Software Engineer at Intel Corporation and a Computer Science graduate student at Arizona State University. He leads a software group responsible for factory planning automation. His research includes modeling and simulation of supply networks, software architecture, and artificial intelligence. He can be contacted by e-mail at <gary.godding@intel.com>.

HESSAM S. SARJOUGHIAN is Assistant Professor of Computer Science and Engineering at Arizona State University, Tempe. His research includes hybrid simulation modeling and intelligent agents, collaborative modeling, distributed co-design, and software architecture. His industrial experience has been with Honeywell and IBM. For more information visit <www.acims.arizona.edu>.

KARL G. KEMPF is Director of Decision Technologies at Intel Corporation and an Adjunct Professor at Arizona State University. His research interests span the optimization of manufacturing and logistics planning and execution in semiconductor supply chains including various forms of supply chain simulation. He can be contacted by e-mail at <karl.g.kempf@intel.com>.