Proceedings of the 2014 Winter Simulation Conference
A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, eds.

DEVELOPING COMPOSED SIMULATION AND OPTIMIZATION MODELS USING ACTUAL SUPPLY-DEMAND NETWORK DATASETS

Soroosh Gholami
Hessam S. Sarjoughian

Arizona Center for Integrative Modeling & Simulation
Computing, Informatics & Decision Systems Engr.
Arizona State University
Tempe, AZ 85281, USA

Gary W. Godding
Daniel R. Peters
Victor Chang

Supply Chain Intelligence & Analytics Group
Intel Corporation
Chandler, AZ, 85226, USA

ABSTRACT

Large, fine-grain data collected from an actual semiconductor supply-demand system can help automated generation of its integrated simulation and optimization models. We describe how instances of Parallel DEVS and Linear Programming (LP) models can be semi-automatically generated from industry-scale relational databases. Despite requiring the atomic simulation models and the objective functions/constraints in the LP model to be available, it is advantageous to generate system-wide supply-demand models from actual data. Since the network changes over time, it is important for the data contained in the LP model to be automatically updated at execution intervals. Furthermore, as changes occur in the models, the interactions in the Knowledge Interchange Broker (KIB) model, which composes simulation and optimization models, are adjusted at run-time.

1 INTRODUCTION

Simulation methods are used for modeling and simulation of supply-demand networks for predicting operations schedules and costs. For example, optimizing large-scale semiconductor supply-chain systems is very complex and the number of configurations of their processes and parameters are many. Therefore, there has been interest in using optimization modules for optimizing the operation of simulation. In this paper, we consider an actual semiconductor manufacturing supply-demand system (aka supply-chain system). Efficiency in the management of such systems can reduce costs by many millions of dollars each year (Wu, Erkoc and Karabuk 2005, Kempf 2004).

Semiconductor supply-demand networks are a collection of suppliers, inventories, processes, transportations, and customers (Kempf 2006). Raw material is processed in sequential and parallel stages to produce many different kinds of products to customers. This enterprise can be divided to manufacturing processes and decision planning parts. The key variables of interest for discrete processes and logistics include stochastic processing times in building products in multiple stages and in different geographies. Another important variable is inventory holdings across manufacturing plants and logistics stages. Both the scale and complexity in manufacturing requires complex decision making, often supported with linear programming where reduced cost and just-in-time (raw material, semi-finished goods, packaging, etc.) delivery across the supply-demand is highly sought after. To simulate the operation of such dynamic enterprises, we can use discrete event modeling specifications such as Discrete Event System Specification (DEVS) and optimization modeling such as Linear Programming (LP) methods. Two important factors in creating realistic models of such enterprise systems are scale and model integration. Manufacturing has many tens of processes and inventories with alternative source to target routes and shipping modes. Similarly, logistics has many inventories and hubs for transportation and delivery to customers in different geographies. Simplifying generation of these simulation models is attractive. Considering the optimization

model, it uses Bill-of-Materials (BOMs) and the manufacturing/logistics state information to forecast desired inventory holdings and shipping routes for materials and products. The optimization model requires an abstract model of the manufacturing/logistics supply-chain. The LP model needs to find an optimal network configuration of the supply-chain having hundreds of thousands of possible flow paths. Therefore, automatic generation of these possible networks is useful. Furthermore, a Knowledge Interchange Broker (KIB) model composing DEVS and LP model has a few hundred data transformations that must be executed at time intervals the simulation and optimization models interact (Huang, et al. 2009). The KIB concept and its use in supply-demand networks are described in previous work (Sarjoughian 2006, Huang, et al. 2009).

In this paper, we describe our research for creating (instantiation, parameterization, and initialization) of DEVS and LP from actual data, supporting the integration of these models with the Knowledge Interchange Broker (KIB) model which interacts with the DEVS and LP models, and performing experiments using these models. A database containing actual Intel's supply-demand network dataset is integrated into the Optimization, Simulation, and Forecasting (OSF) platform (Sarjoughian, Smith, et al. 2013). We note that forecast modeling is excluded in this work. In model creation we take on the problem of automatically generating manufacturing/logistics processes and formulating decision plans using the database and the generated simulation model. The work in this paper was delivered as an Industrial Case Study presentation at the Winter Simulation Conference (Gholami, et al. 2013).

2 RELATED WORK

The significance of automatically generating simulation and optimization models for the entire supply-demand network for the production of semiconductor is well documented (Missbauer and Uzsoy 2011, Fordyce, et al. 2011). Automatic model generation is invaluable for large, complex supply-demand networks where many hundreds to thousands of products are manufactured in dynamical settings and stages under varying demand forecasts. One of the early works is IMPreSS (Leachman, et al. 1996), an automated production planning and delivery quotation system. A more recent work uses mixed-integer programming where optimization for a supply-chain process is driven by heuristics (Denton, Forrest and Milne 2006). In another work, discrete-event simulation is used to model manufacturing process and linear programming to optimize the operation of the supply-demand network using heuristics forecast modeling (Sarjoughian, Smith, et al. 2013).

From the standpoint of automatic simulation and optimization model generation, it is useful to generate them automatically. This need has been recognized from the early days of simulation (Oldfather, Ginsberg and Markowitz 1966). Efforts have been made to increase portability, maintenance, and modifiability of models via automatic model generation from databases (Taylor and Taha 1991). If we restrict autonomous instantiation to simulation models (as opposed to application instantiation (Stauber, Ambrose and Rothwein 2003)), there are two common ways to automatically generate simulation models. One is to use a *descriptive model of a system* with pre-built components and then generate simulation models. Second is to generate simulation models by parameterized and compose simulation models from pre-built components using *data collected from an actual system*. As an example of first way, automatic simulation model generation is proposed and exemplified for a shop floor resource model (Son, Wysk and Jones 2003). The source of models for autonomous model generation may be a database or documents supporting ASCII and XML formats.

Automatic model instantiation, parameterization, and initialization which we have developed follows the second way. This is similar to the work in (Jeong and Allan 2004) where simulation models are automatically generated from a dataset belonging to a discrete manufacturing system. In our case, data gathered from the real supply-demand network along with existing models are contained in a set of database tables which are then used to generate coupled simulation models. Actual data is used to parameterize atomic and coupled model components. From the standpoint of automatic simulation model generation, the difference is in the scale and the complexity of the models (i.e., databases). In our work, we use an

optimization model which requires generating network flow graphs for the optimization model. This is achieved using the same database that is used for generating simulation models.

3 SUPPLY-DEMAND SIMULATION AND OPTIMIZATION

In this section, we summarize the effort made to model an actual supply-demand network in DEVS and DEVS-Suite. The modeled network consists of four Fabrications (FAB), two of each: Die Warehouses (DW), Assembly Test Model (ATM), and Customer Warehouse (CW). The network also possesses 9 Vendor Managed Inventories (VMI) and more than 200 customers. A high-level view of the network is depicted in Figure 1.

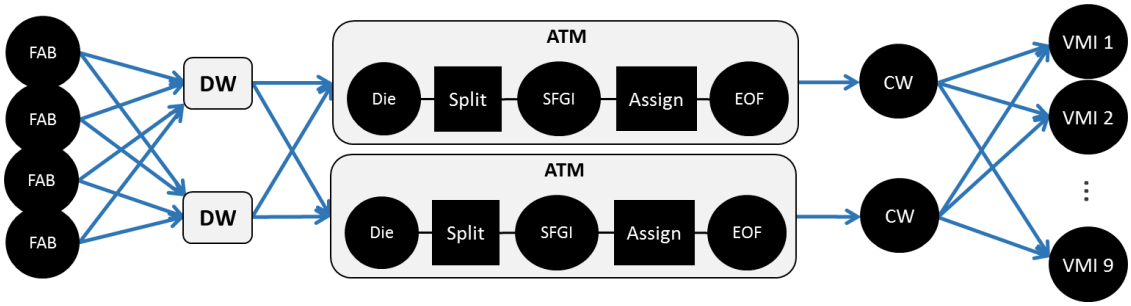

Figure 1: A high-level view of the simulated supply-demand system.

Each facility in the supply chain is modeled as either an atomic or coupled DEVS model. A coupled model may have several atomic/coupled sub-models. Fabrication sites are specified as atomic models which produce *Fab* products. These products are then delivered to DWs (which similar to ATM contain three inventories and processes inside them) in which they are split in order to be processed for different targets (high performance processors, energy efficient, etc.). ATM sites further split the products and then process them into *FG* (Finished Good), ready to be handed to the customers. These finished goods are stored in CWs in big entities and then shipped to VMIs and customers in smaller proportions.

Each of the DW and ATM facilities (containing inventories and processes) are coupled models containing several internal atomic models for themselves. The internal view of the both facilities are presented as coupled models with circles indicating inventories and rectangles indicating processes. All inventory and coupled models have control-in port with which receives commands for releasing products. These commands may come from a database (historic data) or be generated by an strategic controller as is the case in this research. A strategic controller manages the operation of the system by considering the state of the network, the future demands/supplies, and penalty costs. It controls the operation of the network by sending commands to each individual facility. Notice that commands are for inventories. Inventories store products until receiving a command to release them. This is not the case for processes as they immediately release the products that are processed and ready.

Although the high-level view of this network does not seem to be large, however, DEVS adaptation of such a system contains 620 atomic and coupled models and more than 1000 couplings between them. Building such a system statically is not scalable and impractical. Therefore, we leveraged automatic model generation to handle instantiation, initialization, parameterization, and coupling of all these models. This is further explained in the rest of the paper. The overall view of the simulation/optimization system is depicted in Figure 2. The reader can identify 6 major components in this simulation which are further discussed in Section 0.

The **Supply-Demand Network** contains all simulation models and their interactions among one another. Each component has an outgoing port (status-out port) via which state information of the component is sent out. **Decision Connector** is in charge of receiving state information from the simulation models and send them over to the Knowledge Interchange Broker (KIB). Also, it distributes the commands

it receives from the KIB and LP module among simulation components. **Dataset** stores various sorts of information for instantiation, parameterization, initialization, and testing the network. This component is thoroughly introduced in Section 4. **Network Generator** is a data structure which provides structured data to the LP module. The static structure of the network is formed into a graph and sent to the LP along with dynamic information such as the current stock, demand (in the horizon of one several months), etc. **LP/CPLEX Optimization Module** is designed and implemented using CPLEX, this module receives the input graph (network generator) and optimizes it (find a solution) which maximizes the profit and minimizes the penalty of missing customer demands. The **Knowledge Interchange Broker** works as a mediator as mentioned earlier. The interaction between different parts of the system is modeled and implemented in the KIB. These models, in addition to the type conversion, provides time synchronization, aggregation, disaggregation, broadcasting, etc.

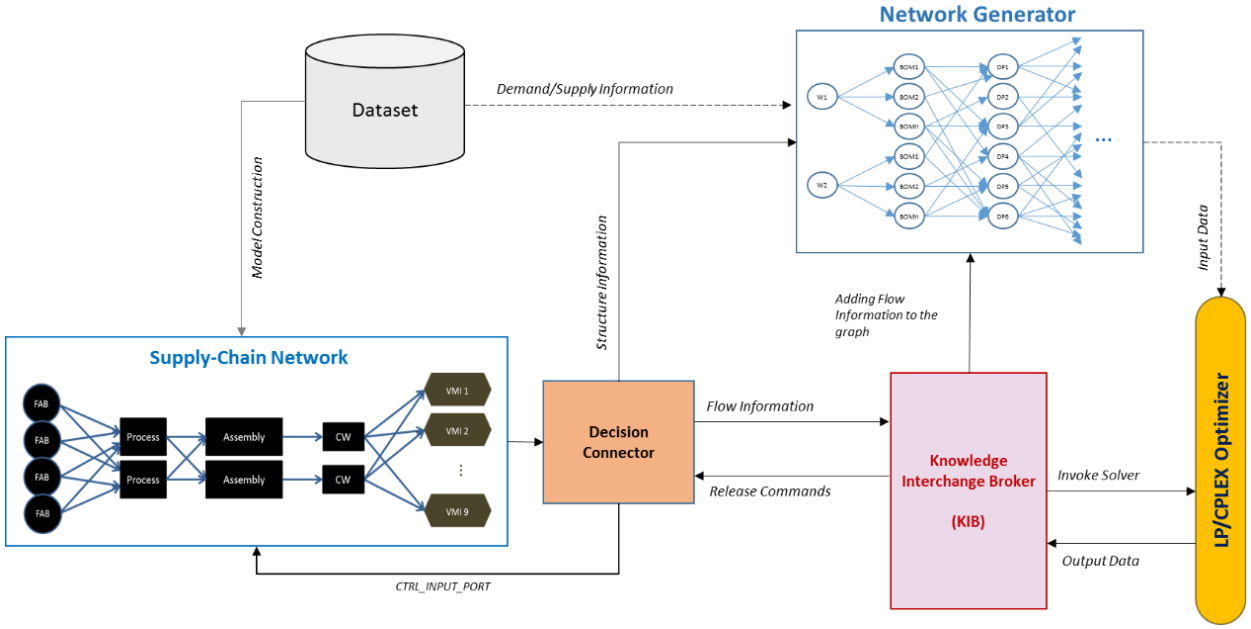

Figure 2: An illustration of the integrated simulation, optimization, and interaction models.

3.1 Issues and considerations in supply-demand modeling and optimization

Although Figure 1 may not show the actual size of the simulation model, as described in Section 6 it is quite large. Instantiation of a model with this size is tricky especially when scalability and modifiability are of importance. Here, we need a scalable method for instantiating, parameterizing, and initializing the network which reads the structure and parameters from a database or a configuration file and construct the network in run time. For company level supply-demand problems with several hundred components and changing topology (from one experiment to the other) we incorporated a similar method. In our case, the network is constructed from a dataset and is dynamically parameterized and initialized after instantiating its components (see Section 6).

Second, is the formulation of the optimization problem from network state variables. In order to optimize the operation of the network based on network parameters and state variables, one has to find a suitable formulation which can be automatically created and passed on to the LP side. On the reverse direction, the optimized operation should be translated back to the supply-demand domain as control commands. For this purpose, we need a data structure which is created at run-time, stores all structure and state information of the network, and passes them on to the LP module (see Section 0).

As real processes have stochasticity in their operation, the DEVS models must present the same level of stochasticity to accurately imitate the behavior of the physical system. Since we aimed at realistic modeling of the network, accurate behavior of each site/process/inventory was investigated from real historic data and embedded into the model. The stochasticity is formulated as mathematical distribution functions and incorporated as processing time or shipping time for related components.

Finally, validation of the simulation models should be taken very seriously. If the simulation diverges from the plan (expected amounts) it could be because of the errors in the simulation module or the incorrect commands that come from the control module. Moreover, when such a divergence occurs the gap gets greater over time and it becomes impractical to find the source of the error. Therefore, each simulation model must be extensively validated before being used in the system.

4 SEMI-AUTOMATIC MODEL INSTANTIATION, PARAMETERIZATION, AND INITIALIZATION

Automatic model construction is defined according to the dataset (see Figure 2). The dataset stores the structural configuration and parameterization of the network in several tables which are used by the highest-level coupled model for instantiation and coupling. After that, each model has to be parameterized in terms of processing/shipping times (in terms of distribution functions), penalty costs, etc. The most important parameter is BOM (bill of material) which specifies how products are converted to one another in split and assign processes. In the initialization phase, each component is initialized with products of different types. Among all tables in the database, we focus on tables *Site*, *Intransit_tpt*, *Process_tpt*, *Products*, and *BOM*. Each record in the *Site* table specifies one facility of the network. So, instantiation of simulation components is done with the assistance of this table. The coupling of these facilities are determined by the *Intransit_tpt* table. This table specifies a shipping (which is a component in the simulation model) by recognizing its source/destination facilities and a random distribution function (specified by a string representing the type of the distribution and at most five number parameters) for the shipping time. Based on the type of the distribution, any number of parameters may be used; for example, a triangular distribution uses 3 parameters. The facilities and their connections fall into the category of static structure of the network. For parameterization, in addition to *Intransit_tpt* (which parameterizes shipping components), we have *Process_tpt* and *BOM* which specify the processing times and bill of material (product split and assign processes), respectively. Both of these tables need references to the *Products* table which stores all product names and their stage in the chain. Each record in *Site* is associated with several records in *Process_tpt* which specify the processing times (distribution functions) for each type of product.

For instantiating, several methods have been developed and used. Coupled models such as ATM and DW are constructed using *makeATM* and *makeDW* methods which not only instantiate all inner elements of ATM and DW in the coupled model but also couple them together and connect the *control_in* and *control_out* ports to those of the coupled model. The novelty of this kind of model generation is the level of flexibility it provides. For creating a network with a completely different topology or parameters, one needs only to deal with a database (provided by the company owning the supply chain) which is much more convenient than a flat file or changing the code. In addition, the stages inside coupled models can also be determined from the dataset. In other words, there is no need to define configuration of the internal components of DW and ATM (i.e., composition of model components)as these can be identified from the dataset. It is assumed the dynamics of the model components are well-defined in terms of creating coupled models. As for parameterization of the models, a method goes over all parameter records in the dataset, find its associated model (by name), and set the parameter inside the simulation model.

One benefit of our approach for dynamic model instantiation, parameterization, and initialization is that the topology or parameters can be changed over time. These topology/parameter changes are then dynamically applied to DEVS models (which can be modified at runtime). Therefore, the simulation can change its most fundamental foundations in runtime. Although we are not using such a capability for

supply-demand networks, one can certainly find suitable applications for it. The dataset holds various product types, processing configurations, historic data, and structural information.

5 OPTIMIZATION PROBLEM FORMULATION

In order to transform structure and state information into LP format (for optimization) we require an efficient and dynamic data structure which can be manipulated at every cycle and sent to the LP. The static structure encapsulates all the static characteristics of the network such as topology, BOM, product types, etc. A graph is designed with weights on its arcs to represent the static structure. Later the state information will be added to this graph. In order to provide the structure (static) to the LP, several methods are implemented which encode the network into the graph. Since each facility (FAB, DW, ATM, CW, and Shipping) has its own characteristics and formulation each one of them has its own structure encoder method. The pseudo code presented in Figure 3 are for adding the static structure of those facilities.

```
FOR each PRODUCT in Table PRODUCTS

      AddNode("DEPART", PRODUCT.name)
      AddNode("ARRIVE", PRODUCT.name)
      AddNode("OUT", UP-STREAM-NODE)
      AddNode("IN",DOWN-STREAM-NODE)

      AddEdge("ARRIVE", "DEPART", cost)
      AddEdge("OUT", "ARRIVE")
      AddEdge("DEPART", "IN")

END FOR
```

a) Adding structure for *Shipping*

```
FOR each BOM in Table BOM

      IF BOM.input IS OF TYPE "SFGI" THEN
            inName = "SFGI"
            outName = "OUT"
      ELSE
            inName = "IN"
            outName = "SFGI"
      ENDIF

      AddNodeIfNotExists(inName, BOM.input)
      AddNodeIfNotExists(outName, BOM.output)
      AddNode(BOM.key, input product)

      AddEdge(inName, BOM.key)
      AddEdge(BOM.key, outName, BOM.percentage)

END FOR
```

b) Adding structure for *ATM*

```
FOR each BOM in Table BOM

      AddNodeIfNotExists("in", BOM.input)
      AddNodeIfNotExists("out", BOM.output)
      AddNode(BOM.key, input product)

      AddEdge("in", BOM.key)
      AddEdge(BOM.key, "out", BOM.percentage)

END FOR
```

c) Adding structure for *DW*

```
FOR each PRODUCT in Table PRODUCTS

      IF PRODUCT IS OF TYPE "WAFER"
            AddNode("OUT", PRODUCT.name)
      ENDIF

END FOR
```

d) Adding structure for *FAB*

```
FOR each PRODUCT in Table PRODUCTS

   IF PRODUCT IS OF TYPE "WAFER"
      AddNodeIfNotExists("IN", PRODUCT.name)
      AddNodeIfNotExists("INVENTORY", PRODUCT.name)
      AddNode("OUT", PRODUCT.name)

      AddEdge("IN", "INVENTORY")
      AddEdge("INVENTORY", "OUT")
   ENDIF

END FOR
```

e) Adding structure for *CW*

Figure 3: Adding static structure for Shipping, ATM, DW, FAB, and CW.

For all components in the network except FAB, we need two nodes as *in* and *out*. These nodes are connected to upstream and downstream nodes via *Shipping*. In ATM, DW, and CW the two nodes are connected to a middle node. Furthermore, ATM and DW specify the percentage of product conversion on the arc. Figure 4 provides a visual example on how the structure information of a DW is transformed into the graph. This process receives wafers as input products (W_1 or W_2) and outputs six types of dies. For splitting the wafers it has *n* different BOMs (or recipes). Therefore, a graph is generated with two starting nodes (W_1 and W_2), each of the connected to *n* different BOMs. The BOMs are then connected to the output products based on their split configuration. For example, consider BOM_1 which converts

W_1 to DP_1, DP_3, and DP_4 in 40%, 40%, and 20% proportions, respectively. So, BOM_1 node is connected to DP_1, DP_3, and DP_4 with their split percentage set as the weight of the outgoing arc (the thick lines).

The dataset we are currently using contains 460 product types, 400 BOMs, 800 various processing times, 250 sites, 340 shipping elements, and more than 50000 demands for a full year. Translating this into a optimization problem (using the approach described above) results in extremely large graphs, each containing few hundred thousand nodes. The entire graph is handed to the CPLEX/LP module for optimization.

The state information (current stock, currently under process, supply, and demand for the next 30 days) are added to the graph as weighted arcs between nodes. Supply information are added to the outgoing arc from FAB to DW. Similarly, current stock in every inventory is added as weighted arc between the inventory and the following process/inventory. Therefore, the addition of state information to the graph does not change the structure of the graph; instead it adds new arcs to it.

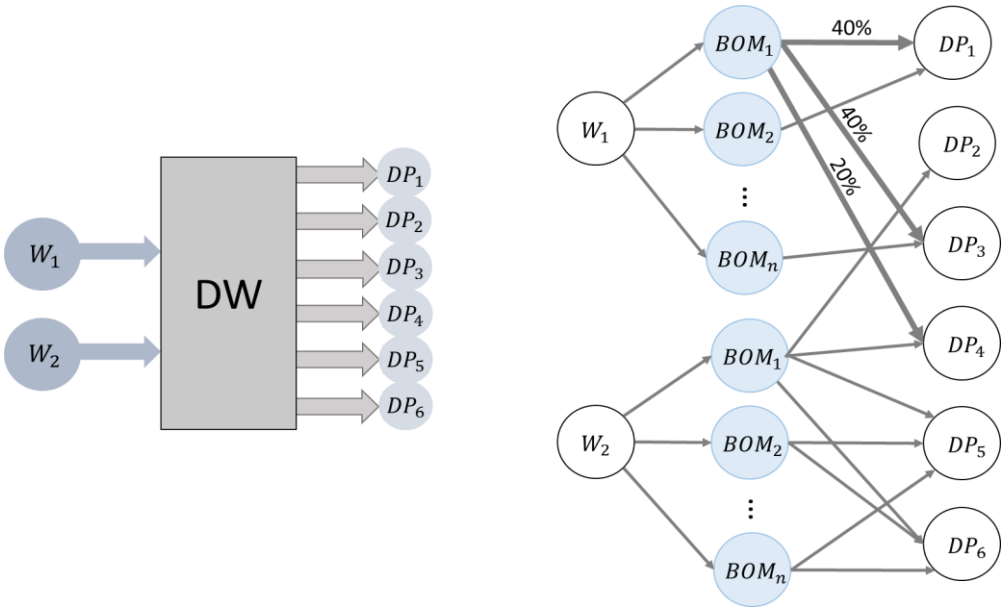

Figure 4: Transformation of a process with input and output products with BOMs into a flow graph.

6 SIMULATION MODEL AND OPTIMIZATION MODULE INTEGRATION

Our ultimate product in this research is an integrated simulation/optimization platform in which the optimization module manages the operation of the network. For this purpose, automatic simulation model instantiation and parameterization was designed and developed. Also, formulating the LP problem from the parameters and state variables of the simulation was another problem that was addressed in this research. This optimization module receives general information from simulation components and optimizes their operation by sending control commands to each facility via their *control-in* port. In addition to the simulation and optimization modules, a mediator must model and manage the interaction between the two. This module is called Knowledge Interchange Broker (KIB).

The discrete event simulation contains supply-demand models (e.g. Fabrication, Assembly Test, Customer, Warehouse, etc.). The optimization module (e.g. control strategy) receives information such as the supply, customer demand, current stock of each inventory, penalty costs, etc. as input and optimizes the operation of the supply-demand network by running a CPLEX optimization program on the state information it received. Commands are generated and sent back to the simulator to control the operation. All these interactions have to go through the KIB. The KIB receives state and structure information from

the simulation module and formulates it as an LP problem. In Figure 2, we can identify six major components for this integrated environment. The supply-demand network simulation, automatic model construction via the dataset, and the optimization module (along with the network generator) are described in Sections 3, 4, and 5. In addition to these, the Decision Connector component is in charge of receiving state information from the simulation models and sending them to the KIB. Also, it distributes the commands it receives from the KIB among simulation components. It has logic to aggregate or disaggregate state information and commands when needed. The concept of the KIB comes from poly-formalism in which the KIB (as a separate formalism) defines and manages interactions between two other formalisms. Each interaction specifies how to transform data from one model type to another model type. Some or all interactions are executed with respect to time and under a control scheme instructing the order in which any two distinct model types can execute with respect to one another. With all these components implemented and integrated with each other, our platform is ready. In the following section selected experiments conducted using this platform are described.

7 EXPERIMENTATION

We conducted 3 sets of validating experiments on this platform: Single-chain, historic data, and system-wide. In the single-chain experiment, we intend to test facilities under the most simplistic scenarios to make sure their logic is implemented correctly. In historic data experiment, we validate the operation of the simulation models by testing and comparing them with historic data. We expect to see rough consistency between the output of our simulation and the historic data. Finally, in the system-wide experiment, all components of the system are included and we carry out end-to-end testing. The results are provided below. Please notice that due to the sensitivity of the data, details are taken out (product names, site names, quantities, and timestamps).

7.1 Single-chain validation

To simplify the simulation as much as possible, we manually modified the dataset to instantiate a single chain of facilities (one of each Fab, Process, Assembly, CW, and VMI) with shipping between them. Also, we made all processing and shipping times deterministic. In addition to all these, this experiment is done in a simulation standalone mode; meaning there is no optimization, KIB, or network generator. All commands are read from the database with stochasticity removed.

Then, the experiment is based on a single demand forecast and supplying enough raw material for meeting the demand. We make sure that the demand is supplied on time and the flow of semi-finished and finished goods in the chain occurs at right timestamps and with correct amounts. In the single chain supply network there is only one customer and one of each other facility. Also, a clock module synchronizes the time between all simulation models and an experimental frame (EF) collects data from all models. We validated the basics of our simulation model with this experiment and observed that all events happen at the right time, quantities are as expected, and the product split and conversions are done without error.

7.2 Historic data validation

In this experiment, we include stochasticity in the simulation model but without integrating it with the LP and the KIB models. Instead, even the release commands are read from the dataset (regardless of the current state). Also, for this experiment, the complete network is used instead of a single chain as in the previous example. The results are then gathered using a EFG and reported in by-product by-site basis. These results are compared with the expected outcome (the data gathered from the real world again in the dataset) and can be reported in terms of plots as is done in Figure 5.

We expected to see small difference in the plots (as it can be seen in the left plot) because of the stochasticity in shipping and processing times. We gathered the data for all products and sites for a year of

simulation (with historic data records for one year) and compared the results. Similar to the examples shown in Figure 5.

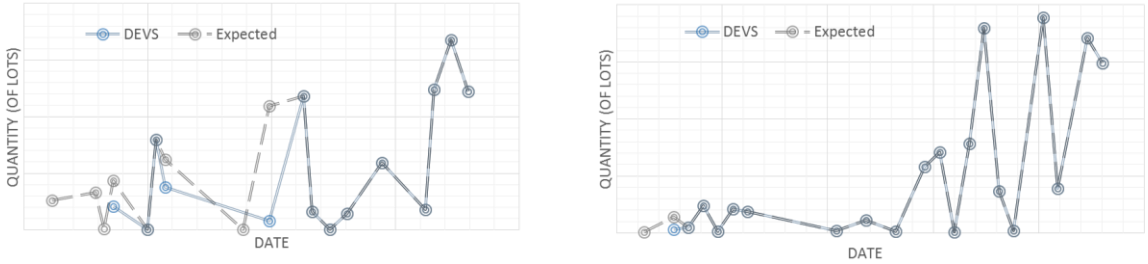

Figure 5: Validating simulated outputs measured against historical data by product and site .

7.3 System-wide validation

In this set of experiments, we used the system shown in Figure 2. Instead of receiving release commands from the dataset, state information is sent to the LP side (via the KIB) and then release commands are generated and sent to the simulation models. For such system, several validation scenarios were necessary. In one set of experiments, the overall operation of the system is test by comparing LP expectations and simulation outcomes. Results for all of these experiments are provided below.

KIB validation: the KIB is validated by conducting several experiments with and without the KIB and comparing their results. Validation of the KIB serves to show that the data transformations, timing and scheduling are correct. In a simulation without the KIB, the transformation of data and time is done manually (not scalable and static). Among those conducted we include one of them (specific to one product and one site) here (see Figure 6). The cumulative view is presented in the top right corner.

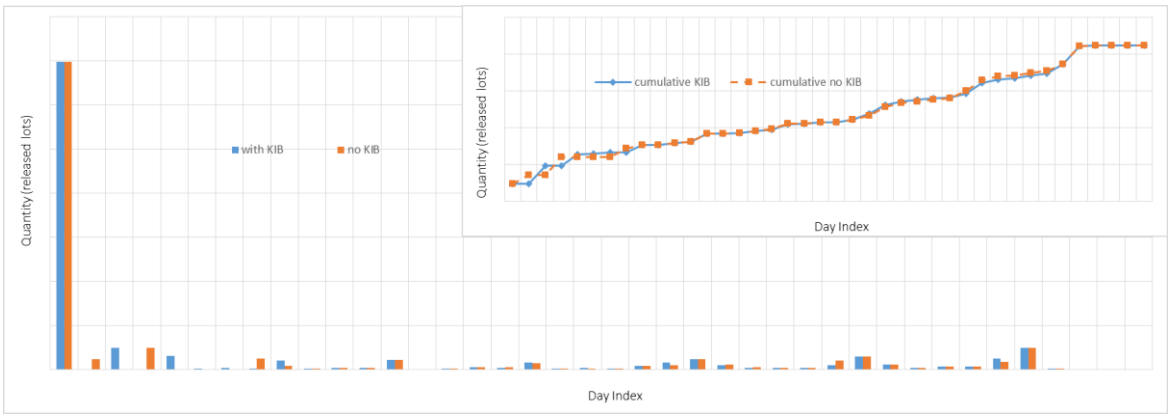

Figure 6: Per period and cumulative release quantities for product *x* at site *y* with and without the KIB.

Overall operation of the platform: in this set of experiments, we compare the actual output with the excepted output of the LP. This would give us a rough estimate of how the system components work together and whether the harmony required to be established by the KIB exists in the system or not. Again, the simulation was executed for a full year and extensive amount of data was gathered. The data is then

analyzed both with *R* and *Excel* to compare LP expectations and simulation outcomes. Figure 7 is the plot generated from our data.

Figure 7 shows the expected and actual release of product *x* at site *y* and their absolute difference (top right corner). This experiment, along with many more, demonstrates that the overall simulation/optimization platform has the required accuracy to be used in practice.

We developed this simulation environment depicted in Figure 2 in Java, using DEVS-Suite 2.1.0 simulator, IBM ILOG CPLEX Optimization Studio 12.5, and Microsoft SQL Server 2012. All the above experiments are carried out on 64bit OS using Java 7. The platform uses Windows 7 with a 2.9GHz Intel Core 2 Duo processor and 8GB of DDR3 physical memory. The minimum, average, and maximum execution times for simulating one week are 8.61, 62.29, and 28.30 seconds. The total execution time for 59 weeks is 131.8 minutes.

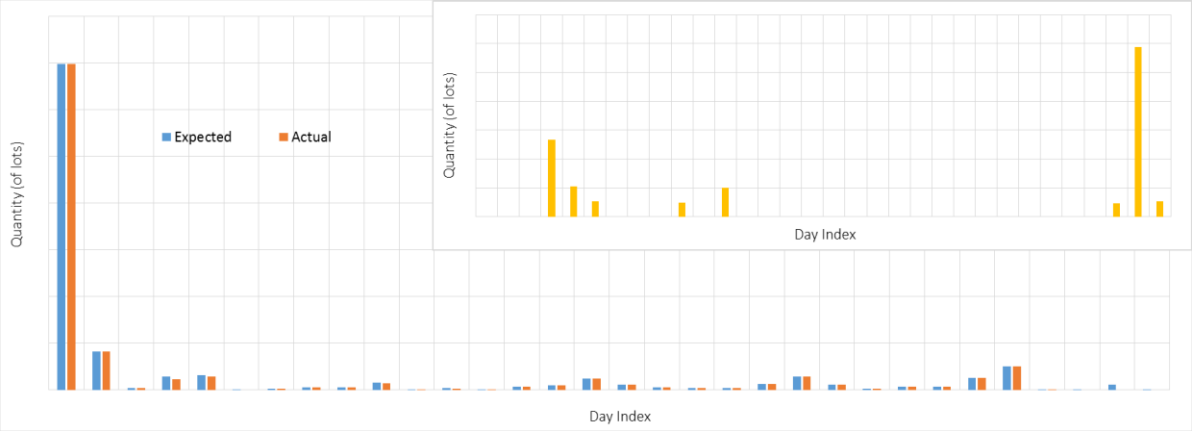

Figure 7: Simulation output (actual) vs. optimized expectation (expected); Absolute difference between simulated output vs. optimization expectation.

8 CONCLUSION

Developing realistic models for supply-demand systems is time consuming. This is particularly true when the goal is to produce high-fidelity simulation studies based on fine-grain characteristics and dynamics of discrete processes and logistics. Equally important to such studies are use of actual decision models that can optimize dynamics of manufacturing over long time horizons. A well-known, common challenge for both of these modeling efforts is scale of the system. This severity of this problem can be reduced if the models can be generated with help from data collected from an actual enterprise. Assuming we have access to models developed for the atomic parts of the manufacturing supply-demand system from which data is available, then very large system-wide models can be automatically generated. We extended the DEVS-Suite simulator with algorithms and database connectivity to generate system-wide simulation models from actual data. Although such models are created prior to simulation, this is not necessary provided changes in the topology of the supply-demand is available in the database. For the optimization model, algorithms were developed to generate product flow networks. This capability is important as it allows the optimization model to account for changes in Bill-of-Materials that can occur, for example, in weekly intervals. Regularly updated LP models may reduce computational time for finding optimal inventory holdings and shipping routes. Finally, bi-directional interactions between DEVS and LP models via the KIB model is also updated which can improve efficiency of the simulation experiments. Current research includes extending our simulation/optimization platform to support customer demand forecasting (Sarjoughian, Smith, et al. 2013) with diagnostics capability.

ACKNOWLEGEMENT

We express our thanks to the anonymous referees including Dr. Ken Fordyce. Their comments and suggestions helped improve the focus of this paper. This research is supported by Intel Research Council.

REFERENCES

Denton, B. T., J. Forrest, and R. J. Milne. 2006. "IBM solves a mixed-integer program to optimize its semiconductor supply chain." *Interfaces* 36 (5): 386-399.

Fordyce, K., C. T. Wang, C.-H. Chang, A. Degbotse, B. Denton, P. Lyon, R. J. Milne, R. Orzell, R. Rice, and J. Waite. 2011. "The ongoing challenge: creating an enterprise-wide detailed supply chain plan for semiconductor and package operations." In *Planning Production and Inventories in the Extended Enterprise*, by K. G. Kempf, P. Keskinocak and R. Uzsoy, 313-387. New York: Springer.

Gholami, S., H. S. Sarjoughian, G. W. Godding, V. Chang, and D. Peters. 2013. "Independent Verification & Validation of Integrated Supply-Chain network Simulation and Optimization Models." *Winter Simulation Conference-Industrial Case Presentation.* edited by R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill, and M. E. Kuhl. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Huang, D., H. S. Sarjoughian, W. Wang, G. W. Godding, D. E. Rivera, K. G. Kempf, and H. Mittelmann. 2009. "Simulation of semiconductor manufacturing supply-chain systems with DEVS, MPC, and KIB." *Semiconductor Manufacturing, IEEE Transactions on* 22 (1): 164-174.

Jeong, K. Y., and D. Allan. 2004. "Integrated system design, analysis and database-driven simulation model generation." *Proceedings 37th Annual Simulation Symposium.* IEEE. 80-85.

Kempf, K. G. 2004. "Control-oriented approaches to supply chain management in semiconductor manufacturing." *Proceedings of the American Control Conference.* IEEE. 4563-4576.

Kempf, K. G. 2006. "Complexity and the Enterprise." *International Conference on Potentials of Complexity Science.* Collegium Budapest. 248-432.

Leachman, R. C., R. F. Benson, C. Liu, and D. J. Raar. 1996. "IMPReSS: An automated production-planning and delivery-quotation system at Harris Corporation—Semiconductor Sector." *Interfaces* 26 (1): 6-37.

Missbauer, H., and R. Uzsoy. 2011. "Optimization Models of Production Planning Problems." In *Planning Production and Inventories in the Extended Enterprise*, by Karl G Kempf, P Keskinocak and R Uzsoy, 437-508. New York: Springer.

Oldfather, P. M., A. S. Ginsberg, and H. M. Markowitz. 1966. *Programming by questionnaire: How to construct a program generator.* RAND Report RM-5129-PR, Santa Monica.

Sarjoughian, H. S. 2006. "Model composability." *Proceedings of the 38th conference on Winter Simulation Conference.* edited by L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto. Monterey, CA. 149-158. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Sarjoughian, H. S, J. Smith, G. W. Godding, and M. Muqsith. 2013. "Model composability and execution across simulation, optimization, and forecast models." *Proceedings of the Symposium on Theory of Modeling & Simulation-DEVS Integrative M&S Symposium.* Society for Computer Simulation International.

Son, Y. J., R. A. Wysk, and A. T. Jones. 2003. "Simulation-based shop floor control: formal model, model generation and control interface." *IIE Transactions* 35 (1): 29-48.

Stauber, C., J. Ambrose, and T. M. Rothwein. 2003. Application instantiation based upon attributes and values stored in a meta data repository, including tiering of application layers objects and components. USA Patent US Patent 6,574,635. June 3.

Taylor, R. B., and H. A. Taha. 1991. "Automatic generation of a class of simulation models from databases." *Proceedings of the 23rd conference on Winter simulation.* Washington, DC, 1209-1217. edited by

Barry L. Nelson, W. David Kelton, Gordon M. Clark. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Wu, D., M. Erkoc, and S. Karabuk. 2005. "Managing capacity in the high-tech industry: A review of literature." *The Engineering Economist* 50 (2): 125-158.

AUTHOR BIOGRAPHIES

SOROSOH GHOLAMI is a Computer Science PhD student at ASU. He can be contacted at <soroosh.gholami@asu.edu>.

HESSAM S. SARJOUGHIAN is an Associate Professor of Computer Science at ASU. He can be contacted at <sarjoughian@asu.edu>. H. S. Sarjoughian is the point of contact.

GARY W. GODDING is an Engineering Manager at Intel Corporation. He can be contacted at <gary.godding@intel.com>.

DANIEL R. PETERS is a Technologist at Intel Corporation. He can be contacted at <daniel.r.peters@intel.com>.

VICTOR CHANG is a Senior Software Engineer at Intel Corporation. He can be contacted at <victor.chang@intel.com>.