Proceedings of the 2006 Winter Simulation Conference
L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.

INTEROPERATING SIMULATIONS OF AUTOMATIC MATERIAL HANDLING SYSTEMS AND MANUFACTURING PROCESSES

Boon Ping Gan
Lai Peng Chan

Planning and Operation Management Group
Singapore Institute of Manufacturing Technology
71 Nanyang Drive
Singapore 638075, SINGAPORE

Stephen John Turner

Parallel and Distributed Computing Centre
Nanyang Technological University
Block N4, Nanyang Avenue
Singapore 639798, SINGAPORE

ABSTRACT

To perform a high fidelity simulation study on a 300 mm wafer fabrication plant, modeling of the manufacturing process (MP) alone is not sufficient. Inclusion of the automated material handling system (AMHS) model is necessary due to the high degree of factory automation. There isn't, however, a single tool that is capable of modeling both the AMHS and MP with sufficient accuracy and granularity. A commercial simulation package such as *AutoMod* is usually used to model the AMHS while *AutoSched AP* is usually used to model the MP. These packages can be integrated using the supplied interoperation module but flexibility in optimizing the execution performance for different simulation models is lacking. In this paper, we present an approach to interoperation based on the High Level Architecture standard. We note that the typical characteristics of disparity in the models' time granularity and frequent model interactions are the obstacle to good execution performance.

1 INTRODUCTION

Various commercial off-the-shelf simulation packages, such as *AutoSched AP* (Brooks 2001a), *AutoMod* (Rohrer 2000), *FlexSim* (FlexSim 2006), and *WITNESS* (Lanner 2006) are available for the modeling of semiconductor manufacturing. These are generally preferred to mathematical models which do not give sufficient accuracy due to the complexity of the semiconductor manufacturing process. Semiconductor manufacturing involves wafers going through a series of layering, patterning, doping, and heat treatment steps, repeating these steps through stages of the manufacturing process. Re-entrant flows, time constraints, varying product mixes, running prototypes and ad-hoc resource breakdowns result in a high degree of variability. Discrete-event simulation is usually the best candidate to portray this dynamic and high variability behavior. The simulation model generates an artificial history of the operations, and is used to study the impact of different policies or capacity changes to the overall performance.

In the past two years, major semiconductor manufacturers such as TSMC, UMC, and Chartered Semiconductor have built 300 mm fabs. TSMC has recently announced that the company will build two more 300 mm fasb with an investment of close to US$6.2 billion dollars. One critical factor that dictates the operational efficiency of a 300 mm fab is the effectiveness of its fab automation. Properly designed and well implemented automation solutions can help to improve the fab cycle time and productivity. The automation solutions include the automated material handling system (AMHS) for moving and storing production lots (interbay and intrabay), real-time advanced equipment/process control, and integrated yield management system. The investment on these integrated solutions can be in the range of US$130 million to US$180 million dollars.

Due to the criticality of the automation solutions and their tight coupling to the efficiency of the manufacturing process, a high fidelity simulation study of a 300 mm fab must include the modeling of both the automation solutions (we are referring to the AMHS) and the manufacturing process. There isn't, however, a single commercial simulation package that is capable of modeling both aspects of the fab with sufficient accuracy and granularity. Our years of experience in simulation have helped us to identify the two most appropriate commercial simulation packages for this purpose, namely *AutoMod* to model the automation solutions and *AutoSched AP* to model the manufacturing process. These simulation packages are supplied by Brooks Automation, and are interoperable through the use of the supplied model communication module. However, the inefficiency of the communication module results in a sequential execution of the two simulation models. This inefficiency can potentially be resolved through the adoption of the High Level Architecture (HLA) standard.

The HLA is an IEEE standard that facilitates interoperability and reusability of simulation components. Commercial simulation packages can be made interoperable at the infrastructure (or execution) level through the adoption of this standard. But it is important for us to realize that interoperation at the infrastructure level alone does not guarantee a valid integrated model. Semantic interoperation of the simulation components is still crucial. In this paper, we will put our focus on resolving the issues of interoperation between *AutoMod* and *AutoSched AP*. We have ensured that the automation and manufacturing process models are semantically interoperable.

One critical hurdle in integrating the automation and manufacturing process models is the disparity in their time granularity: seconds for the automation model and minutes for the manufacturing process model. This disparity results in close to sequential execution of the two models. Wang, Xu, and McGinnis (2005) proposed a time compensation scheme to improve the execution parallelism. But the proposed scheme does not take lots dispatching and random events such as tool down into consideration. Such assumptions restrict the application of the scheme for general simulation models.

Another critical hurdle are the frequent interactions between the manufacturing process and the automation models. The AMHS is required for every production lot movement from one machine to the next. The frequency of interactions could potentially degrade the performance of simulation execution. In this paper, we do not attempt to address these two hurdles. We focus, instead, on the extent to which the two hurdles affect the performance of simulation execution.

This paper is organized as follows: In Section 2, we discuss how *AutoMod* satisfies the HLA-based interoperation requirements and the corresponding mechanism by which interoperation was realized. This is then followed by a discussion on the interoperation between *AutoSched AP* and *AutoMod* in Section 3. A general automation and manufacturing process model is then described in Section 4. Using this model, we studied the contributing factors to the execution inefficiency of the interoperation mechanism in Section 5. Lastly, we conclude our study with an outline of future work in Section 6.

2 INTEROPERATING AUTOMOD USING HLA

2.1 The Interoperation Framework

The Commercial off-the-shelf Simulation Package Interoperability Product Development Group (CSPI-PDG 2006) endorsed by the Simulation Interoperability Standards Organization (SISO) has devised a framework for the interoperation of commercial off-the-shelf simulation packages (CSP). The primary objective of the CSPI-PDG is to create a set of standard reference models, data exchange standards and generic interfaces (Taylor, Turner, and Low 2005) that supplement the HLA standard. The framework simplifies the interoperation efforts through a middleware approach as shown in Figure 1. The middleware (including the *DSManager*) facilitates the interaction between the CSP and the Runtime Infrastructure (RTI) of the HLA (Wang et al. 2004). It implements the interactions protocol for the reference models, transparent time synchronization mechanisms and simulation entities management. This middleware is general for all types of CSPs that satisfy the five following requirements for interoperation devised by Gan et al. (2005a):

- (R1) Ability to initialize the distributed simulation prior to simulation execution.
- (R2) Ability to suspend the simulation execution
- (R3) Access to the time of the next event to be simulated.
- (R4) Ability to introduce new events/entities from the external source into the event list.
- (R5) Access to information of simulation objects/entities that are shared among federates.

Using this interoperation framework, the efforts for interoperating CSPs were reduced significantly.

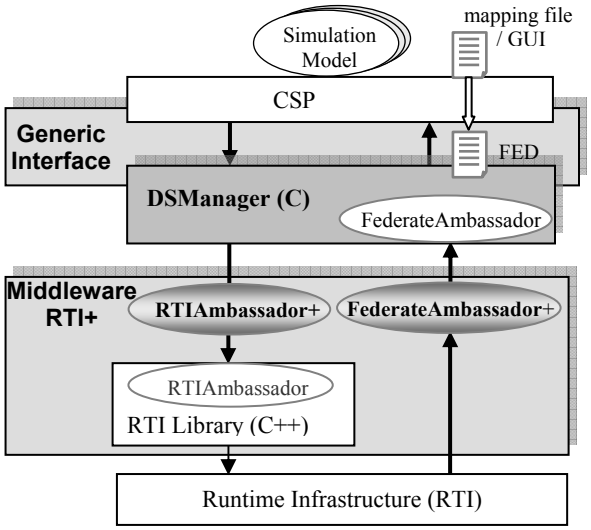

Figure 1: The Interoperation Framework

2.2 Adapting *AutoMod* Using the Interoperation Framework

To adapt *AutoMod* for interoperation using the defined framework, we studied if it satisfies the five requirements. We found that *AutoMod* exposes its simulation execution through the *AutoMod* runtime object. The runtime object

triggers system or user-defined events based on the occurrence of defined conditions in the simulation. A corresponding function implemented in Visual Basic is then executed to handle the event. In addition, the runtime object allows the state of the simulation to be accessed from an external interface. This two-way communication mechanism, as illustrated in Figure 2, facilitates the integration of the interoperation framework, and the fulfillment of the five interoperation requirements. Figure 3 shows the system architecture of the interoperating *AutoMod*.

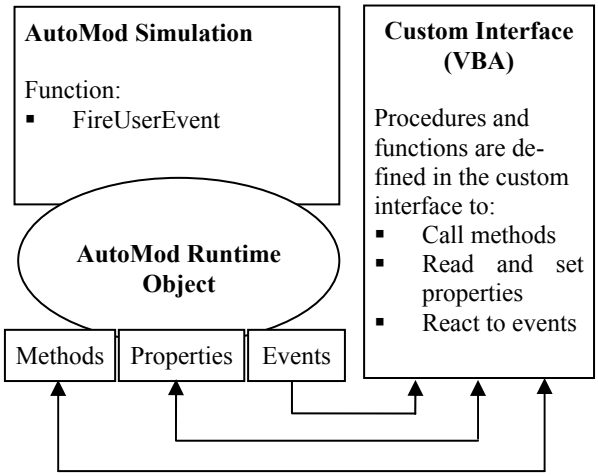

Figure 2: Interaction between Custom Interface and *AutoMod* Runtime Object

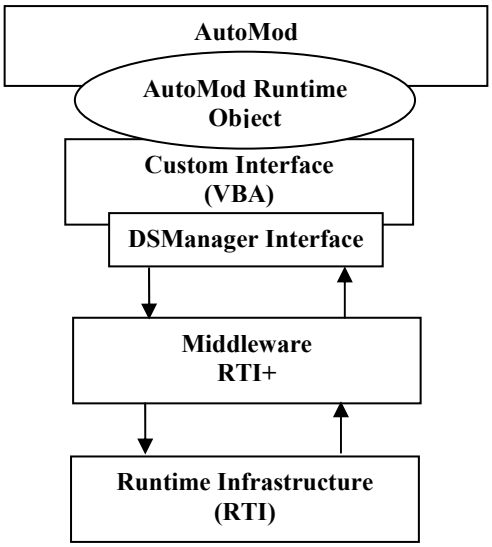

Figure 3: System Architecture of the Interoperating *AutoMod*

(R1) Ability to initialize the distributed simulation prior to simulation execution

The *OnModelReady* event of the *AutoMod* runtime object is triggered after all the initialization of the model is completed and before simulation begins. A function is associated with this event to initialize the distributed simulation prior to simulation execution.

(R2) Ability to suspend the simulation execution

When a user event is triggered by calling the *FireUserEvent* method, the execution control is handed over to the custom interface. This suspends the execution of the simulation and allows the middleware to take appropriate actions for distributed simulation execution. Termination of the simulation is associated with the *OnStateChange* event.

(R3) Access to the time of the next event to be simulated

The *model nextclock* function is called after all the events at the current simulation time are processed and before the simulation clock advances to the next future event. A time synchronization user event is triggered in this function using the *FireUserEvent* method. The minimum of the earliest future event time (fetched using the *FEvFirstTime* function) in the *AutoMod*'s event list is used as the request time to the interoperation framework.

(R4) Ability to introduce new events/entities from the external source into the event list

The *CallFunction* method allows the custom interface to call user-defined functions in the model during simulation. This mechanism is used to introduce new events/entities from an external source to the simulation. Note that the new event/entity could possibly have a timestamp larger than the current simulation clock. A *wait for* primitive is thus issued, with the difference of the event's timestamp and the current simulation clock as the parameter, before the event/entity is handled. This ensures that the event/entity is handled at the correct simulation time.

(R5) Access to information of simulation objects/entities that are shared among federates

The information of simulation objects/entities are defined as attributes, which are accessible to the model. To share the information, the values of attributes are packed into a string and passed as a parameter to the *FireUserEvent* method. The corresponding event handler is then used to send out the information through calls to the interoperation middleware.

2.3 Adapting *AutoSched AP* Using the Interoperation Framework

The *AutoSched AP* satisfies the five interoperation requirements as discussed in (Gan et al. 2005a). It was adapted for interoperation using the defined interoperation framework. This earlier effort enables wafer lots to be rerouted from one model to another through the action list associated with each processing step. The action list is used to define a series of methods to be executed when a new wafer lot arrives at a processing step. Instead of using the default action list, we replaced it with one that invokes the *DSManager* at the processing step that reroutes wafer lot to an external model. This new action list first sends the wafer lot out with the receiving time as its timestamp and then deletes the lot from the sending factory. Simulation execution is suspended after processing of safe events (events that would not result in causality violation). A new time request is issued via the *DSManager* for the next safe time before the control is handed back to the simulation engine.

3 INTEROPERATING *AUTOSCHED AP* AND *AUTOMOD*

3.1 Vendor Supplied Interoperation Module

Brooks Automation, the vendor for *AutoSched AP* and *AutoMod*, supplies a model communication module (MCM) and an *AMAP* custom extension that facilitate the interoperation of the simulation packages (Brooks 2001a). The MCM, which is model independent, manages time synchronization and enables message exchange using a network socket between any two models. It supports two time synchronization modes, namely exact synchronization and periodical synchronization. When exact synchronization is used, only one model is active at a time. The active model advances its time to its next future event and completes all the events at the current simulation time. Once this is done, the timestamp of its next future event is compared to the timestamp of the inactive model's next future event. The model with a smaller timestamp for the next future event is made the active model while the other is made inactive. For periodical synchronization, the models complete all events within the same synchronization period in parallel. The next synchronization period is initiated only after all models complete the current period. The period chosen has impact on the model's accuracy. The larger the period, the less accurate the model becomes. As such, efficiency is traded off against accuracy when the periodical synchronization approach is used.

The *AMAP* uses the MCM module to model the movement of wafer lots from one tool to another. It extends the *storage* feature of the *AutoSched AP* such that the *storage system, storage*, and *system travel capacity* are mapped to *path mover system*, *control point*, and *number of vehicles* in *AutoMod* (Brooks 2001a). A request message is sent to the *AutoMod* model for every lot movement from one *storage* to another. The *AutoMod* replies when the lot is picked up or when the lot is delivered. The movement is simulated by considering the distance and the paths to take between two points, traffic congestion, and complex vehicle behavior such as vehicle speed.

One drawback of using the *AMAP* is the inefficiency in message exchange. The request message from *AutoSched AP* to *AutoMod* and the pickup and delivery messages from *AutoMod* to *AutoSched AP* are sent at the instant of time that they occur. This results in two tightly coupled simulation models whose potential parallelism is not fully exploited. This is probably not a critical issue when the MCM module has the limitation of executing both models on one computer. To our knowledge, the only possible means of executing the two models on different computers is to use the MCM plus (Brooks 2003) which does not come as a standard package.

3.2 Interoperation Using the HLA

The approach described earlier for adapting *AutoSched AP* for interoperation can be used for the interoperation of *AutoSched AP* and *AutoMod*. But a simpler way is to modify the *AMAP* custom extension of the *AutoSched AP* such that the modeler is not aware of whether the MCM or our interoperation framework is used. This was done by replacing all the lot sending and receiving methods in the *AMAP* custom extension with the corresponding methods provided by the *DSManager*. Though there was no one-to-one mapping of methods, the process of replacing was straightforward. Using this approach, any models that use *AMAP* can directly be replaced with the modified *AMAP* for interoperation. This provides the benefit of transparency as the modeler does not need to make any changes to existing models that are using the *AMAP* custom extension. The only difference is that instead of invoking the *AutoMod* model automatically when *AutoSched AP* starts, *AutoMod* has to be invoked manually. But the added advantage of the modified *AMAP* is the possibility of executing models on different computers.

3.3 Time Synchronization Issues

In our previous work on interoperating *AutoSched AP* models, an optimized time synchronization algorithm using the manufacturing process flow was devised to reduce the number of interactions between the simulation execution and the middleware (Gan et al. 2005a and Gan et al. 2005b). This had the effect of improving the simulation execution time by a factor of ten compared with a conventional time synchronization algorithm.

This optimized time synchronization algorithm is not able to resolve the problem of execution inefficiency due to the disparity in time granularity between the *AutoSched AP* and *AutoMod* models. The benefits of the optimized time synchronization algorithm can only be fully exploited when a small number of processing steps trigger external events. But the manufacturing process and the automation models do not posses such characteristics. In fact all processing steps in the manufacturing process trigger an external event as movement of production lots from one step to another is handled by the automation system. As this paper is focusing on identifying the different means available to interoperate *AutoSched AP* and *AutoMod* models, the time synchronization issue will not be addressed here.

4 SIMULATION MODEL

To evaluate the performance of interoperation between a manufacturing model in *AutoSched AP* and an automation model in *AutoMod*, we used a simplified model provided by the vendor. In this model, the *AutoSched AP* drives the *AutoMod* simulation where lots are generated only at the *AutoSched AP* model. Two different product types are released into the *AutoSched AP* model with a constant interval. There are a total of 72 processing steps for each product type with 6 different station families. Messages are sent to *AutoMod* to move lots from one location to another. The *AutoSched AP* model determines all the destinations for moving a lot in the simulation, including the lot's final destination and each of its intermediate steps. The *AutoMod* model holds the responsibility of determining the path on which a lot travels and which vehicle it takes to reach its next stop. The retrieval time, travel delays and delivery time are determined based on conditions in the *AutoMod* model, such as vehicle congestion, vehicle velocities and distances in the system.

When a move request message is sent by *AutoSched AP*, *AutoMod* creates a new lot at the starting location and determines how long it will take to retrieve the lot and move it to the destination location. On successfully retrieving a lot, *AutoMod* responds to *AutoSched AP* that the lot has been picked up. In the meantime, the lot is delayed in the *AutoSched AP* model until it has successfully been delivered to its destination. On reaching its intended destination location, the lot is destroyed in the *AutoMod* model. *AutoMod* then responds to *AutoSched AP* by sending a message to indicate that the lot has been delivered so that lot processing at the *AutoSched AP* model can resume. The storages in *AutoSched AP* are modeled in *AutoMod* as shown in Table 1.

Figure 5 illustrates the layout of the automated material handling system. Lots from one system can travel to another through bridges. An *AutoMod* vehicle drops the lot at a bridge control point in the first system and another vehicle picks it up at the same bridge control point in the next system. Each control point located in the path mover system corresponds to its storage in the *AutoSched AP*. Next to each control point is a label with three numbers that represent the following:

- 1st digit: The total number of lots claiming the control point (including lots that have claimed but are not yet at the storage).
- 2nd digit: The total number of lots occupying the control point and waiting to leave.
- 3rd digit: The total number of lots currently claiming and occupying the control point and not yet waiting to leave. A more detailed description on this model can be found in Brooks (2001b).

Table 1: Mapping of *AutoSched AP* to *AutoMod* Models

AutoSched AP	***AutoMod***
Storage system	Path mover system with the same name as storage system
Storage	Control point with same name as storage
System travel capacity	Number of vehicles in system

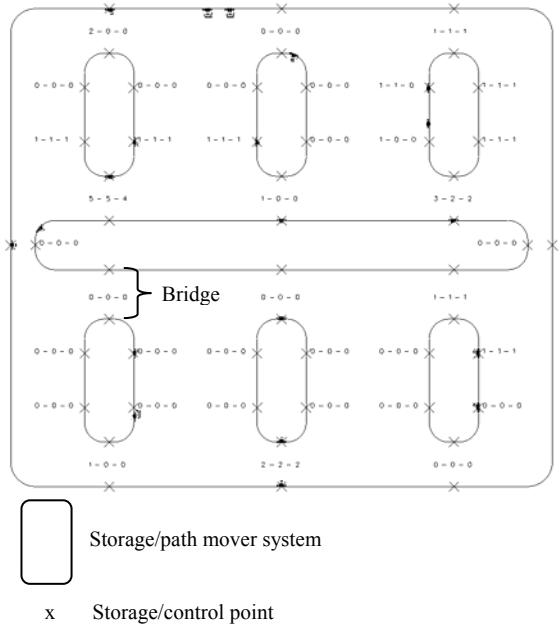

Figure 5: Layout of Automated Material Handling System

5 EXPERIMENTAL RESULTS

Two factors that could possibly restrict the execution efficiency of interoperating *AutoSched AP* and *AutoMod* models are: (1) disparity in time granularity, and (2) frequent (zero time increment) interactions between the two models. The first factor can be observed by measuring the number of time requests issued by each simulation package. A significant deviation in this performance measure implies a

significant disparity in time granularity. The average time step size, computed by taking the simulation run length divided by the number of time requests, can also be used as an indicator. The intensity of interactions between the two models can be observed by measuring the ratio of external events to internal events, termed as event ratio hereafter. The higher the event ratio, the higher the frequency of interactions. This generally infers a tightly coupled model with less opportunity for further optimization.

Using the simulation model described in Section 4, experiments were conducted to measure four performance metrics: number of time requests issued, average time step size, ratio of external to internal events, and execution time. Simulation was repeated for different work-in-progress (WIP) levels (12, 18, and 24 lots) to evaluate the scalability of the interoperating *AutoSched AP* and *AutoMod* models. The scalability was studied as the performance of our interoperation approach is sensitive towards the WIP level. The higher the WIP, the higher the frequency of time requests. This usually has negative (but unavoidable) impact on the speed of simulation execution.

Table 2 and 3 show the number of time requests issued and the average time step size for the *AutoSched AP* and *AutoMod* models. As we can see, *AutoMod* issues more than ten times the number of time requests as compared to *AutoSched AP*. A time request is issued for less than one second (simulation time) interval consistently. On the other hand, *AutoSched AP* issues time requests with a larger interval, in the range of tens of seconds. This disparity in time granularity can significantly slowdown the simulation as no model parallelism can be exploited.

Table 2: Number of Time Requests Issued

WIP Level (Lots)	No of Time Requests	
	AutoSched AP	*AutoMod*
12	14249	150889
18	20237	239443
24	26444	348817

Table 3: Average Time Step Size

WIP Level (Lots)	Average Time Step (seconds)	
	AutoSched AP	*AutoMod*
12	12.1	1.1
18	8.5	0.7
24	6.5	0.5

Table 4: Event Ratio for the Interoperating *AutoSched AP* (*ASAP*) and *AutoMod* (*AM*) Models

WIP Level	No of External Events		No of Internal Events		Event Ratio	
	ASAP	*AM*	*ASAP*	*AM*	*ASAP*	*AM*
12	4943	9886	14250	150890	0.35	0.07
18	7641	15282	20238	239444	0.38	0.06
24	10463	20926	26445	348818	0.40	0.06

Table 4 shows that the external event to internal event ratio for the *AutoSched AP* and *AutoMod* models is approximately 0.4 and 0.06 respectively. This implies that there are enormous opportunities to exploit the model parallelism as time requests only need to be issued for those events that potentially could trigger external events. In the best case, the number of time requests should be equal to the number of external events sent by a model. Nevertheless, this best case is not attainable as it would require complete knowledge of the simulation execution. A more practical approach is to devise a scheme that predicts the time at which an external event will be sent. Some knowledge of the model characteristics is essential to devise such a scheme. This same concept has been applied in our earlier work to improve the execution efficiency of interoperating *AutoSched AP* models (Gan et al. 2005b).

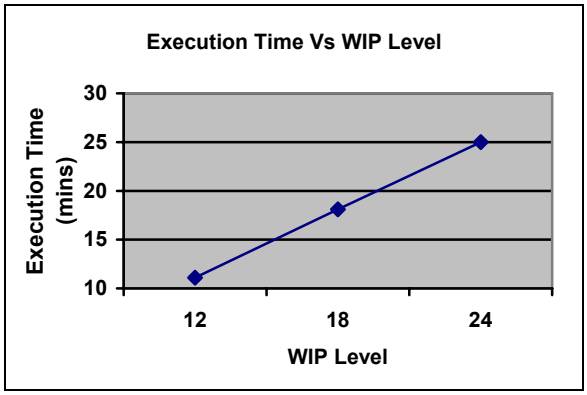

Figure 6: Execution Time for the Interoperating *AutoSched AP* and *AutoMod* Models

The relationship between the execution time and the WIP level is linear as shown in Figure 6. This increase in execution time is generally due to the larger number of events being processed for higher WIP level (Table 4). Our future work aims to bring down the simulation execution time by exploiting the enormous opportunities that are presenting in these models.

6 CONCLUSIONS

The interoperation of commercial off-the-shelf simulation packages remains a challenge even with the availability of standards such as the CSPI-PDG and the HLA. The simulation execution time is one critical hurdle that cannot be addressed by any standard. Analysis has to be done to identify if opportunities exist for exploiting the underlying model parallelism. Often, this requires understanding of the model characteristics in order to devise an efficient mechanism.

Nevertheless, we have shown this task of interoperating commercial off-the-shelf simulation packages is not impossible. Our work in interoperating *AutoMod* and

AutoSched AP has made it possible to simulate 300 mm wafer fabrication plants with high fidelity. Adopting the standard defined by CSPI-PDG helped to reduce the efforts in the interoperation exercise. The issue of execution efficiency remains to be addressed, however. This is not an unsolvable problem as our analysis has revealed that there are opportunities to improve the simulation execution time. We are currently working on a customized time synchronization algorithm which exploits the model characteristics. In addition, the issue of zero time increment interactions will also be addressed.

REFERENCES

Brooks Automation. 2001a. AutoSched AP User's Guide v7.0.

Brooks Automation. 2001b. AutoSched AP Customization Guide v7.0.

Brooks Automation. 2003. Model Communications User's Guide: A Module for AutoMod.

CSPI-PDG. 2006. <http://www.cspi-pdg.org> [accessed June 1, 2006]

Flexsim. 2006. <http://www.flexsim.com/software/flexsim> [accessed March 2, 2006].

Gan B. P., M. Y. H. Low, S. J. Turner, X. Wang, and S. J. E. Taylor. 2005a. Interoperating *AutoSched AP* Using the High Level Architecture. In *Proceedings of the 2005 Winter Simulation Conference,* ed. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, 394-401. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Gan B.P., M. Y. H. Low, X. Wang, and S. J. Turner. 2005b. Using Manufacturing Process Flow for Time Synchronization in HLA-Based Simulation. In *Proceedings of the Ninth IEEE International Symposium on Distributed Simulation and Real-Time Applications*, 148-157.

Lanner Group. 2006. <http://www.lanner.com/products/simulation_suite/witness.php> [accessed March 2, 2006].

Rohrer, M. W. 2000. AutoMod Tutorial. In *Proceedings of the 2000 Winter Simulation Conference*, ed. J.A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, 170-176. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Taylor S. J. E., S. J. Turner and M. Y. H. Low. 2005. The COTS Simulation Interoperability Product Development Group (CSPI-PDG). In *Proceedings of the 2005 European Simulation Interoperability Workshop*, 05E-SIW-056.

Wang, K., S. Xu, and L. F. McGinnis. 2005. Time Management in Distributed Factory Simulation, A Case Study Using HLA. In *Proceedings of the 2005 Winter Simulation Conference,* ed. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, 1781-1786. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Wang, X. G., S. J. Turner, M. Y. H. Low and B. P. Gan. 2004. A Generic Architecture for the Integration of COTS Packages with the HLA. In *Proceedings of the UK Operational Research Society Simulation Workshop*, 225-233.

ACKNOWLEDGMENT

The authors would like to thank Mr. Gao Junjie for providing support in exploring the interoperation mechanism for the *AutoMod* simulation package.

AUTHOR BIOGRAPHIES

BOON PING GAN is a Senior Research Engineer with the Planning and Operations Group at the Singapore Institute of Manufacturing Technology. The focus of his research is on the application of distributed simulation technology for supply chain simulation. He received a Bachelor of Applied Science (Hons) in Computer Engineering and a Master of Applied Science from Nanyang Technological University of Singapore in 1995 and 1998, respectively. His research interests are parallel and distributed simulation, parallel programs scheduling, and application of genetic algorithms. His e-mail address is <bpgan@SIMTec.a-star.edu.sg>.

LAI PENG CHAN is a Senior Research Engineer with the Planning and Operations Management Group at the Singapore Institute of Manufacturing Technology. She obtained her BSc and MTech at the National University of Singapore. Her research interests include modeling and simulation, optimization and intelligent systems. Her e-mail address is <lpchan@SIMTech.a-star.edu.sg>.

STEPHEN JOHN TURNER joined Nanyang Technological University (Singapore) in 1999 and is currently an Associate Professor in the School of Computer Engineering and Director of the Parallel and Distributed Computing Centre. Previously, he was a Senior Lecturer in Computer Science at Exeter University (UK). He received his MA in Mathematics and Computer Science from Cambridge University (UK) and his M.Sc. and Ph.D. in Computer Science from Manchester University (UK). His current research interests include: parallel and distributed simulation, distributed virtual environments, grid computing and multiagent systems. His e-mail address is <assjturner@ntu.edu.sg>.