Proceedings of the 2005 Winter Simulation Conference
M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.

ADVANCED DECISION LOGIC IN SIMULATION OF MATERIAL FLOW PROCESSING NETWORKS

Douglas A. Bodner
Ke Wang
Sheng Xu

School of Industrial & Systems Engineering
765 Ferst Drive, N.W.
Georgia Institute of Technology
Atlanta, GA 30332, U.S.A.

ABSTRACT

Material flow processing networks are ubiquitous in modern society. Such networks embody uncertainty, advanced decision logic, and increased performance expectations. As such, they have been subject to analysis by a variety of methods, with one of the most prominent being discrete-event simulation. Simulation enables detailed flow modeling, incorporates uncertainty and allows experiment-based performance assessment. Traditional approaches to simulation are not well-suited to modeling advanced decision logic, though. This paper explores the issue of representing advanced decision logic and presents a reference model for material flow processing networks to support such representations. Implementation issues are discussed, as well.

1 INTRODUCTION

Material flow processing networks are ubiquitous in modern society. Broadly considered, they include factories, warehouses, supply chains, hospitals and service centers. These types of systems experience considerable uncertainty, including demand variability and system component failures. With advanced tracking and computation available, more complex decision logic typically is being implemented in these systems, to address uncertainty and also to improve performance. Performance improvement is increasingly a concern, where performance may be stated in terms of throughput, cycle times, work-in-process level, or threats addressed.

One widely used tool in analyzing and designing material flow systems is discrete-event simulation. Widely applied in manufacturing for over thirty years (Smith 2003), simulation increasingly is used to study supply chains, service systems, health care systems, and emergency response networks. Traditional approaches to simulation are well-suited to modeling material flow through a system's network of resources, as well as representing uncertainty through random number generation from a wide availability of probability distributions. However, traditional approaches to simulation are not as well-suited to representing the advanced decision logic increasingly found in material flow networks. Advanced decision logic includes such functions as aggregate planning, optimizing vehicle routings, rescheduling operations, deadlock avoidance, or performing an iterative evaluation. Such representations are critical to being able to prototype the decision logic increasingly found in material flow systems.

This paper discusses representations needed for advanced decision logic in simulation models of material flow processing networks. The remainder of this paper is organized as follows. Section 2 discusses issues involved with simulation and the representation of advanced decision logic. Section 3 presents a reference model to support modeling advanced decision logic. Section 4 describes issues involved in design of decision logic in simulation. Section 5 presents implementation results to date. Section 6 concludes with thoughts on future research.

2 SIMULATION AND ADVANCED DECISION LOGIC

Simulation software typically is based on one of three "world-views," or classes of modeling formalisms, including process-interaction, event-scheduling and activity scanning (Law and Kelton 2000). These formalisms are ways of organizing and representing the fundamental elements of a simulation model – events, activities, decisions and information.

The most widely used is the process-interaction world-view, in which simulation entities travel through a set of blocks, whereby they seize, hold and release resources, and they are routed through a network of such resources. This world-view corresponds quite well with the physical characteristics of a material flow processing network, where material queues for various resources, is processed, and

then is moved to the next resource. In addition, this worldview facilitates model-building, as the modeler need only specify entities and their types, as well as the block structure for resources. The events and activities implied by these modeling constructs are provided for the modeler automatically. This representation is not as well-suited to modeling complex decision logic. Mujtaba (1994) and Platzman and Gershwin (1986) discuss the difficulty of representing active decision-making in computer integrated manufacturing systems. Ruiz-Meir and Talavage (1989) discuss integrating artificial intelligence to overcome the difficulty in modeling complex decision logic.

On the other hand, the event-scheduling world-view focuses on events that may occur in the system being modeled. Events must be explicitly modeled and coded. While this provides much flexibility, including the capability to represent advanced decision logic, it does not provide modeling discipline needed to do so. Thus, the modeler must code such decision logic, in what may wind up as an ad-hoc process. In any event, the event-scheduling approach entails a significant amount of coding.

Finally, the activity scanning world-view focuses on activities in the simulation, and the resources that serve as pre-requisites to activity initiation. It is useful for modeling systems in which many resources must come together for the various activities in the system modeled. The main drawback is the computation required, since the state of the system must be scanned continually to determine whether prerequisite conditions are in place for an activity to start. For this reason, activity scanning is not widely used in material flow system modeling.

The limitations of these three world-views have led to several bodies of research. The first centers on use of the object-oriented paradigm to encapsulate decision-making in objects (Adiga and Glassey 1991; Mayrand, Lefrancois, and Montreuil 1993; Mize et al. 1992; Narayanan et al. 2003; Ulgen and Thomasma 1990; Zeigler 1990). As discussed by Narayanan et al. (1998), detailed object-oriented simulation involves extensive effort, at least in part due to the event-scheduling approach taken in most of this research. Unfortunately, little of this research has impacted the commercial simulation software community.

Another body of research uses the process-interaction formalism to model advanced decision logic (Bodner, Govindaraj, and McGinnis 2000; Van der Zee 2003). These efforts seek to develop formalized models of decision logic using process interaction modeling blocks, through which control entities are routed to represent procedural decision logic in response to events and states in the physical flow system. Still, the object-oriented approach allows for more flexibility in modeling (Narayanan et al. 2000).

Open source simulation toolkits, such as Silk (Kilgore 2003) and DSOL (Jacobs, Lang and Verbraeck 2002) provide another potential source for representing advanced decision logic. While they do not provide pre-specified tools for representing such logic, they enable the modeler to extend the existing modeling constructs to do so. What is needed, though, is the modeling discipline to enable such representations to be effective across a variety of domains.

Recent efforts are investigating multi-formalism architectures to address the inclusion of advanced decision logic in simulation models. Godding, Sarjoughian, and Kempf (2004) describe integrating discrete-event simulation process models with linear programming decision models under a model composibility framework. Venkateswaran, Son, and Jones (2004) present an architecture that integrates discrete-event simulation with system dynamics to address production planning issues. The research in this paper seeks to integrate process interaction simulation with event-scheduling, though the focus in this paper is on decision logic representation within simulation models.

3 MATERIAL FLOW REFERENCE MODEL

This research seeks to specify discrete-event simulation representations for advanced decision logic, while at the same time taking advantage of the material flow representation characteristics of the process interaction world-view. As a first step, as in all modeling, it is necessary to describe formally the domain being modeled. Software engineering provides methods for this under the rubric of domain analysis (Arango and Prieto-Diaz 1990, Sodhi and Sodhi 1999). In manufacturing, there has been a body of work to specify reference models, or canonical descriptions of a class of systems under study for purposes of modeling (Biemens and Vissers 1989, Kim 2004). In addition, several efforts have undertaken specification of classification schemes for production systems (MacCarthy and Fernandes 2000; Schmitt, Klastorin, and Shtub 1985).

This research draws upon previous work in modeling decision logic in manufacturing systems (Bodner, Govindaraj, and McGinnis 2002; Narayanan et al. 2003). While useful, this previous work has limitations that drive enhancements to it.

- The modeling constructs are overly manufacturing-specific, instead of being suitable for the larger class of material flow processing networks.
- A limited set of decision capabilities is specified, mainly relating to shop floor control.
- The process-interaction representation of decision logic relies on WAIT-SIGNAL blocks (from ARENA®). While useful and powerful, widespread use of these blocks often is discouraged by expert simulationists due to their complexity.
- Decision logic is purely reactive, i.e., it cannot schedule an event.

This section presents an overview of work toward a formal reference model, using domain analysis. Material

flow systems are characterized by material, locations in which material resides, movement of material between locations, decision logic units, information and operators. The term "decision logic unit" or "DLU" refers to a decision-maker within the simulation (e.g., controller, scheduler, etc.).

Material in many cases is passive, i.e., it is moved and processed by other elements in the system. In some instance, material is active, for example when considering customers as material in a service system or health system. Being active implies some decision-making capability inherent in the material, e.g., decide where to go next. Material also can be categorized as being a substrate or components, in the case of assembly/disassembly systems and warehousing systems (i.e., pallets and cases). Figure 1 shows a categorization scheme for material.

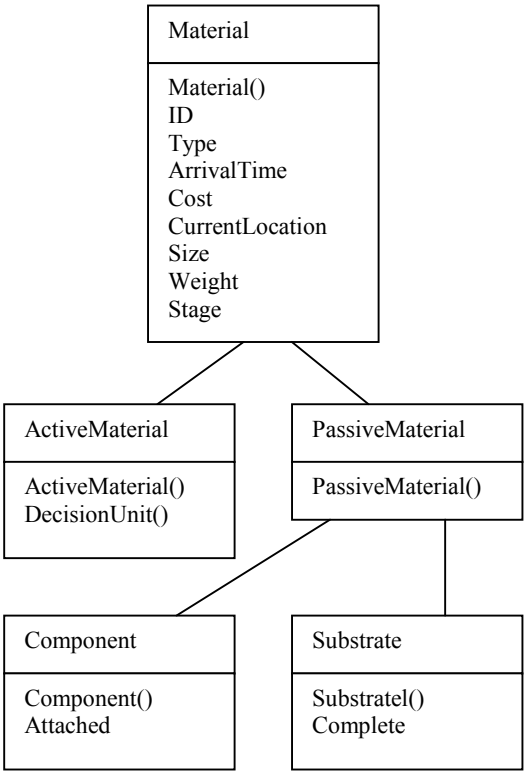

Figure 1: Material Categories

Locations typically are conceptualized as a material flow network. The basics of this network are processing locations that transform material in some way (nodes) and material flow arcs between them. Such a network may also contains vehicles that traverse arcs (and that have their own locations where material resides), conveyors, and storage locations. In practice, the network may be decomposed into several constituent networks, each having its own method to move material between locations.

Decision logic units are responsible for sequencing activities, selecting actions from among multiple alternatives, distinguishing items to select based on an evaluation, scheduling future events or activities, receiving feedback from the physical elements of the system, and issuing commands to the physical elements of the system. Decision methods can be in the form of simple numerical heuristics, optimizations, iterative evaluation routines, etc.

Decision logic units operate on the basis of information about the physical system and about goals of the material flow system. Information about the physical system consists of (i) information about its elements, and (ii) information about relationships between elements. Element information consists, for example, of current location of material items, current status of processing machines and current location of vehicles. It also consists of descriptive behavior (e.g., a machine has a certain pattern of up-times, down-times, setup-times), as well as prescriptive behavior (e.g., a manufactured product is required to undergo a certain sequence of processing steps). Relational information informs the decision logic unit, for example, what process locations can perform which operations. It may also inform the decision logic unit which transporters are capable of moving material between a set of locations. A logical queue is relational in the sense that it provides the relationship between material items and their next destination locations. Information on system goals includes such information as performance measurements to be considered (e.g., maximize bottleneck process utilization, maximize throughput, maximize identification of security threats).

Operators combine aspects of (i) material or transporters, in that they move between physical locations in the system, (ii) processing locations, in that they have capability to transform material, and (iii) decision logic units, in that they may engage in independent decision-making behavior. As such, they are not readily modeled by traditional approaches to simulation.

While much of the preceding terminology is manufacturing-oriented, it should be understood that it is applicable to the broader class of material flow systems. Figure 2 illustrates the various aspects of the underlying reference model described here.

4 DECISION LOGIC UNIT DESIGN

In designing a decision logic unit, an approach similar to Tirpak et al. (1992) is taken. They derive a generic architecture for a flexible manufacturing cell, with a material flow network, a material transport device and a controller (decision logic unit). Their interest is to promote this architecture as fractal in nature, i.e., applicable to flexible manufacturing in general, at different levels in a hierarchy. For shop floor control issues in a job shop environment, this representation is adequate.

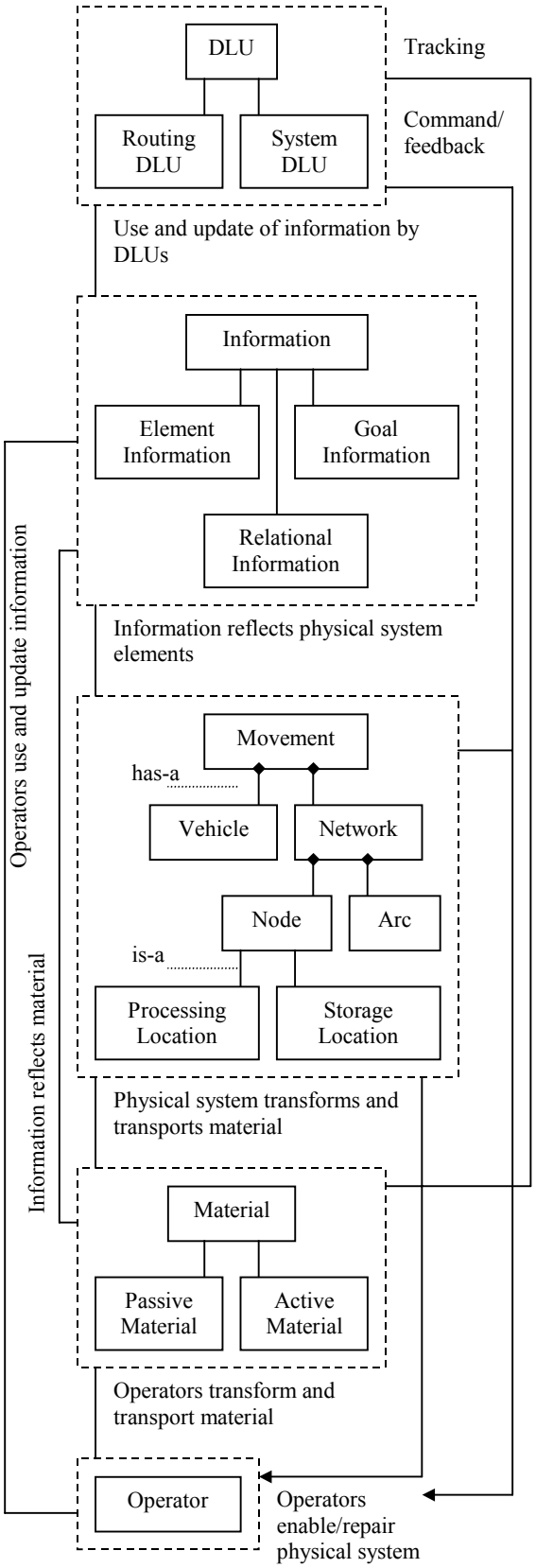

Figure 2: Material Flow System Model

However, for the broader class of material flow systems, which exhibit different flow patterns and different decision-making routines, more functionality from the decision logic unit is required. The approach here, then, is to identify a set of generic decision logic modules, where a module takes into account the following:

- Material flow pattern (e.g., unidirectional multi-stage, re-entrant flow, aggregation, cross-dock);
- Decision problems to be solved (e.g., material release, dispatching, aggregate planning);
- Algorithms to solve decision problems;
- Information needed by algorithms to solve decision problems (bill of materials, material location, machine status);
- Events that trigger a decision or are triggered by a decision (e.g., process completion, machine failure), and
- Control points, or interfaces where the decision logic unit communicates with the physical system (e.g., material at finish point in processing). In between control points, the process-interaction formalism governs material flow.

Further details about these classification parameters are provided in (Bodner 2005). In a model of a particular system, a decision logic unit interacts with a specified subset of the physical flow system, for example, a material flow network within the overall system network. It could be that one of the modules captures all the functionality needed for the decision logic unit in question. It could also be that multiple modules are combined in one decision logic unit to achieve the functionality required. For example, a small warehouse may have disaggregation material flow (pallets converted to cases, which are shipped as orders) and light assembly (items from cases assembled into kits). Thus, the DLU structure must accommodate multiple modules.

From a technical perspective, the DLU must have the following capabilities, as shown in Figure 3:

- Communication with other DLUs and the physical flow system through a standardized protocol,
- Logic processing that refers incoming communication to the appropriate generic controller function (e.g., status update) or module, and
- Decision logic embedded in modules.

5 EXAMPLE DLU IMPLEMENTATIONS

This section briefly presents three example implementations of decision logic units, based on three different material flow application domains. These implementations use open source Java™ simulation toolkits for material flow modeling, either Simjava (Howell and McNab 1998) or

DSOL (Jacobs, Lang and Verbraeck 2002). The physical flow system is modeled via process-interaction, while the decision logic governing the system is captured in the DLU, which is implemented using the Java language.

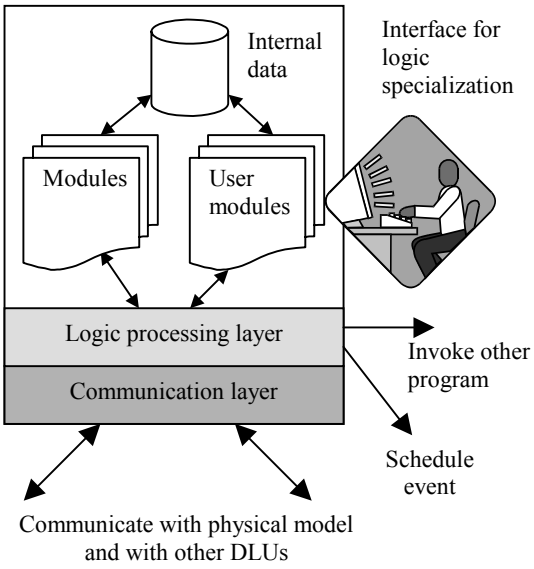

Figure 3: Decision Logic Unit Functions

The first model centers on a five-machine mini-fab environment for semiconductor manufacturing (Kempf 1994). Here, semiconductor wafers follow a six-step process plan, which is executed on five machines. Routing flexibility is allowed, since four of the process steps can be performed by two different machines. The decision logic module is very characteristic of shop floor control, except that re-entrant material flow is present. Material induction is handled via CONWIP. Dispatching is handled via first-come-first-served. Machines that perform the same operations are grouped in workstations. Selection from machines in a workstation is done by first-available. Table 1 describes the relevant DLU module characteristics.

The second model focuses on a surgery ward and an accompanying intensive care unit. Here, patients are the material. They arrive to the system and are admitted based on need and operating room time required. This is formulated as a variant of a knapsack problem, where need is the item value, and time required the cost. Capacity constraints involve the number of operating rooms, availability of a room in intensive care afterward, and availability of a doctor for surgery. The DLU invokes CPLEX® via CONCERT® to solve the knapsack-like formulation in determining which patients can be admitted. The DLU sends relevant information, and receives back the optimal answer. While patients are in the intensive care unit, nurses and doctors (as operators) visit them on assigned rounds. Nurses have a preset schedule. Doctors have leeway in their schedule. Table 2 shows the DLU module characteristics.

Table 1: Semiconductor Mini-Fab Characteristics

Material flow	Re-entrant
Decision problems	Shop floor control (induction, dispatching, routing).
Algorithms	CONWIP for induction, FCFS for dispatching, first-available for routing.
Information	Machine status, process plans, current stage and location of wafers, machine operational capabilities, number of wafers in system.
Events	Job created, job arrives to machine, job finishes at machine, job sent to next machine, job leaves system.
Control points	Entry to system, post-processing.

Table 2: Health System Characteristics

Material flow	Linear flow
Decision problems	Patient admission, patient assignment to operating rooms, patient assignment to intensive care rooms.
Algorithms	Knapsack formulation solved by outside software.
Information	Patient need, patient surgery time requirement, room availability, doctor availability.
Events	Patient arrives, patient dispatched to surgery, patient finishes surgery, patient arrives to IC room, patient leaves system, nurses and doctors arrive and leave during rounds.
Control points	Entry to system, post-surgery.

The final application is multi-stage project management. Here, projects are modeled as material that flows through the project management system (i.e., approval and funding for go-ahead). If they receive continued go-ahead and are successful, projects proceed to succeeding stages. In general, there are n stages for a project; in the specific implementation, $n = 3$. Any proposed projects arriving to a particular stage in a particular budget cycle are grouped together, and funding is decided for the group based on a budget constraint for that stage in that budget cycle. Similarly to the hospital example, decisions for funding are

made via a knapsack problem formulation, which is solved via CPLEX.

In the knapsack formulation, item cost is represented by the project's budget request for that stage. The knapsack capacity is, of course, the total budget available for projects at that stage in that budget cycle. The value can be computed in at least two ways. Each project is assumed to have a payoff upon completion of all stages. Therefore, value can be net present value (NPV) of the project's payoff and future budget requests. Alternatively, value can be computed using real options, analogous to financial options, in that a purchase confers the right to decide later whether to purchase the asset/payoff (Trigeorgis 1996). Real options capture the value of flexibility in being able to terminate funding in later stages of a multi-stage project.

The decision to fund the last stage does not have an option value, so its value can be computed via NPV. The decision to fund the second stage can be formulated as a simple option (i.e., option to pursue the last stage later). The decision to fund the first stage can be formulated as an option-on-an-option. A simple option can be valuated using the Black-Scholes method (Black and Scholes 1973). The option-on-an-option can be valuated using the notion of compound options (Geske 1979). Both of these require computation of the cumulative distribution function of the normal distribution. This can be performed in a simulation model by a numerical approximation. The compound option, however, is an iterative evaluation function that also requires solving for the zeros of the original option formula. This is not so easily done. The approach here is to implement the computations in MATLAB®, which is invoked by the simulation model when project valuation is needed. Table 3 provides the relevant DLU module characteristics.

Table 3: Project Management Characteristics

Material flow	Linear flow
Decision problems	Project funding at each stage.
Algorithms	Knapsack formulation solved by outside software.
Information	Budget at each stage for each cycle, project budget requests at each stage, project payoff, stage durations, interest rate, volatility (variation over time of payoff).
Events	Project arrives, project sent to next stage, project discarded, project finishes stage.
Control points	Entry to system, in-between stages.

These example implementations illustrate the notion of modules. Future work involves formalizing this concept and creating a library of such modules for material flow scenarios. Each of these systems, while simple in nature, can potentially have complex decision-making involved. One theme is that tools outside the simulation model are needed to support the decisions to be made. This makes sense, since real systems may use such tools in their operation. One question, obviously, is the computational effect of multiple calls to outside tools during execution of a large-scale simulation model. This issue will be explored in future applications.

6 CONCLUSION AND FUTURE RESEARCH

This paper has presented an approach to representing advanced decision logic in material flow systems. It builds on previous research in modeling decision logic in manufacturing systems. However, it encompasses the broader class of material flow systems, including service systems. Being able to represent advanced decision logic in simulation models is critical to being able to prototype the behavior of this type of decision logic in real systems. This is significant for important performance and safety reasons in today's complex, expensive and safety-critical systems. Future work involves the following:

- Operationalizing the material flow system classification with additional example systems and enhancing as needed;
- Enhancing the decision logic unit representation specified so far;
- Studying the computational effect of the DLU approach vs. traditional simulation methods, and of the use of outside tools such as optimizers;
- Investigating the use of agent-based simulation techniques in providing decision capability to operators in simulation models.

ACKNOWLEDGMENTS

This work has been funded by the National Science Foundation under grant no. DMI-0423360.

REFERENCES

Adiga, S., and C. R. Glassey. 1991. Object-oriented simulation to support research in manufacturing systems. *International Journal of Production Research* 29: 2529-2542.

Arango, G., and R. Prieto-Diaz. 1991. Domain analysis concepts and research directions. In *Domain analysis and software systems modeling*, ed. R. Prieto-Diaz and G. Arango, 9-33. Los Alamitos, California: IEEE Computer Society.

Biemens, F. P., and C. A. Vissers. 1989. Reference model for manufacturing planning and control. *Journal of Manufacturing Systems* 8: 35-46.

Black, F., and M. Scholes. 1973. The pricing of options and corporate liabilities. *Journal of Political Economy* 81: 637-659.

Bodner, D. A. 2005. Hybrid models for simulation-based prototyping of decision logic in material flow systems. In *Proceedings of the 2005 NSF DMII Grantees Conference*. Scottsdale, Arizona: Arizona State University.

Bodner, D. A., T. Govindaraj, and L. F. McGinnis. 2002. PIMSIM: controller-based simulation and prototyping of material flow control systems. In *Progress in material handling research: 2002*, ed. R. Meller, M. K. Ogle, B. A. Peters, G. D. Taylor and J. Usher, 77-92. Charlotte, NC: Material Handling Institute.

Geske, R. 1979. The valuation of compound options. *Journal of Financial Economics* 7: 63-71.

Godding, G. W., H. S. Sarjoughian, and K. G. Kempf. 2004. Multi-formalism modeling approach for semiconductor supply/demand networks. In *Proceedings of the 2004 Winter Simulation Conference*, ed. R. G. Ingalls, M. D. Rossetti, J. S. Smith and B. A. Peters, 232-239. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Howell, F., and R. McNab. 1998. Simjava: a discrete event simulation package for Java with applications in computer systems modeling. In *Proceedings of the First International Conference on Web-based Modelling and Simulation*. San Diego, California: Society for Computer Simulation.

Jacobs, P. H. M., N. A. Lang, and A. Verbraeck. 2002. DSOL: a distributed Java based discrete event simulation architecture. In *Proceedings of the 2002 Winter Simulation Conference*, ed. E. Yücesan, C.-H. Chen, J. L. Snowdon and J. M. Charnes, 793-800. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Kempf, K. 1994. Intel five-machine six step mini-fab description. Intel/ASU Report, Arizona State University.

Kilgore, R. A. 2003. Object-oriented simulation with SML and Silk in .NET and Java. In *Proceedings of the 2003 Winter Simulation Conference*, ed. S. Chick, P. J. Sánchez, D. Ferrin and D. J. Morrice, 218-224. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Kim, H. 2004. Reference model based high fidelity simulation modeling for manufacturing systems. Doctoral. dissertation, School of Industrial & Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia.

Law, A. M., and W. D. Kelton. 2000. *Simulation modeling and analysis*. Boston: McGraw-Hill.

MacCarthy, B. L., and F. C. F. Fernandes. 2000. A multidimensional classification of production systems for the design and selection of production planning and control systems. *Production Planning & Control* 11: 481-496.

Mayrand, E., P. Lefrancois, and B. Montreuil. 1993. An agent-oriented simulation model of manufacturing activities. *Integrated Computer Aided Engineering* 1: 137-146.

Mize, J. H., H. C. Bhuskute, D. B. Pratt, and M. Kamath. 1992. Modeling of integrated manufacturing systems using an object-oriented approach. *IIE Transactions* 24: 14-26.

Mujtaba, M. S. 1994. Simulation modelling of manufacturing enterprise with complex material, information and control flows. *International Journal of Computer Integrated Manufacturing* 7: 29-46.

Narayanan, S., D. A. Bodner, U. Sreekanth, T. Govindaraj, L. F. McGinnis, and C. M. Mitchell. 1998. Research in object-oriented manufacturing simulations: an assessment of the state of the art. *IIE Transactions* 30: 795-810.

Narayanan, S., D. A. Bodner, U. Sreekanth, T. Govindaraj, L. F. McGinnis, and C. M. Mitchell. 2003. Software patterns for modelling discrete-part manufacturing systems using objects. *International Journal of Modelling and Simulation* 23: 29-42.

Narayanan, S., D. A. Bodner, U. Sreekanth, T. Govindaraj, L. F. McGinnis, C. M. Mitchell, and J. Evans. 2000. Modeling a printed circuit board assembly line using objects. *Simulation* 75: 227-240.

Platzman, L. K., and S. B. Gershwin. 1986. Simulating computer integrated manufacturing systems: how to model what traditional methods force you to ignore. In *Proceedings of the 1986 IEEE International Conference on Systems, Man, and Cybernetics*, 1007-1008. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Ruiz-Meir, S., and J. Talavage. 1989. A hybrid paradigm for modeling of complex systems. In *Artificial intelligence, simulation and modeling*, ed. L. E. Widman, K. A. Loparo and N. Nielsen, 381-395. New York: Wiley Interscience.

Schmitt, T. G., T. Klastorin, and A. Shtub. 1985. Production classification system: concepts, models and strategies. *International Journal of Production Research* 23: 563-578.

Smith, J. 2003. Survey on the use of simulation for manufacturing system design and operation. *Journal of Manufacturing Systems* 22: 157-171.

Sodhi, J., and P. Sodhi. 1999. *Software reuse: domain analysis and design process*. New York: McGraw-Hill.

Tirpak, T. M., S. M. Daniel, J. D. LaLonde, and W. J. Davis. 1992. A note on a fractal architecture for modelling and controlling flexible manufacturing systems. *IEEE Transactions on Systems, Man, and Cybernetics* 22: 564-567.

Trigeorgis, L. 1996. *Real options: managerial flexibility and strategy in resource allocation*. Cambridge, MA: The MIT Press.

Ulgen, O. M., and T. Thomasma. 1990. SmartSim: an object-oriented simulation program generator for manufacturing systems. *International Journal of Production Research* 28: 1713-1730.

Van der Zee, D. J. Modeling control in manufacturing simulation. In *Proceedings of the 2003 Winter Simulation Conference*, ed. S. Chick, P. J. Sánchez, D. Ferrin and D. J. Morrice, 791-798. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Venkateswaran, J., Y.-J. Son, and A Jones. 2004. Hierarchical production planning using a hybrid system dynamic-discrete event simulation architecture. In *Proceedings of the 2004 Winter Simulation Conference*, ed. R. G. Ingalls, M. D. Rossetti, J. S. Smith and B. A. Peters, 1094-1102. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Zeigler, B. P. 1990. *Object-oriented simulation with hierarchical, modular models: intelligent agents and endomorphic systems*, San Diego: Academic Press.

AUTHOR BIOGRAPHIES

DOUGLAS A. BODNER is a senior research engineer affiliated with the Tennenbaum Institute and the School of Industrial & Systems Engineering at the Georgia Institute of Technology. He is a member of IEEE, IIE and INFORMS. He can be reached via email at <dbodner@isye.gatech.edu>.

KE WANG earned her BS in Electronic Engineering at ShangHai JiaoTong University, and Ph.D. in Management Engineering at ZheJiang University. She is currently a Ph.D. student at the Georgia Institute of Technology. She is also working as a graduate research assistant at Georgia Institute of Technology, and her current research focuses on modeling and analysis of material flow systems. She can be reached via email at <kwang@isye.gatech.edu>.

SHENG XU earned his M.ENG. in Mechanical Engineering and Automation at Zhejiang University, China, M.ENG. in High Performance Computing at National University of Singapore (the Singapore-MIT Alliance Program). He worked as an Associate Research Fellow/Research Engineer in Singapore Institute of Manufacturing Technology. He also worked as a software engineer, senior program designer, systems analyst etc. for many years in China. He is currently a Ph.D. student at the Georgia Institute of Technology, and his current research focuses on modeling and simulation of manufacturing and logistics systems. . He can be reached via email at <sxu@isye.gatech.edu>.