Proceedings of the 1996 Winter Simulation Conference
ed. J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain

A NEW PARADIGM FOR MANUFACTURING ENTERPRISE MODELING: REUSABLE, MULTI-TOOL MODELING

Dursun Delen
David B. Pratt
Manjunath Kamath

Center for Computer Integrated Manufacturing
School of Industrial Engineering and Management
Oklahoma State University
Stillwater, OK 74078, U.S.A.

ABSTRACT

The power of modeling of manufacturing systems can be enhanced through the application of reusable and pluggable modeling constructs. In this paper we discuss a framework for model reusability and how the seperation concept together with object oriented paradigm support this capability. We then discuss the importance of reusable and pluggable organizational controls in a modeling environment. Several important issues related to the implementation of reusability and pluggability are summarized. Examples are presented to illustrate our approach for implementing reusable and pluggable organizational controls.

1 INTRODUCTION

Advanced manufacturing technology, without a doubt, is a major corporate advantage in today's global market. Strategic application of advanced technology can markedly improve a manufacturer's product quality, responsiveness to customers, process control, process flexibility, and capital investment flexibility, all of which are determinants of global manufacturing competitiveness.

The stochastic nature of manufacturing systems, coupled with ever changing market behavior, makes it difficult for manufacturing engineers to analyze and design/redesign a complete discrete part manufacturing system. It is rarely feasible to do experiments with the actual system because such experimentation is often too costly or too disruptive to the system. In many cases, the "system" might not even exist, as in the case of the design of a new discrete part manufacturing system. Nevertheless, we may want to study it in its various proposed alternative configurations to recommend how it should be built in the first place. For these reasons, it is usually necessary to build a model as a representation of the system and conduct the analysis using this model.

It has been demonstrated many times that modeling, analytical or simulation, is vital for the design of complex manufacturing systems (Leung and Suri 1990; ElMaraghy and Ravi 1992; Mize et al. 1992; Suri and de Treville 1993). Simulation and analytical models are used to analyze the stochastic behavior of these systems under various scenarios. Nevertheless, modeling of such complex systems is not without its shortcomings. First, models have traditionally been viewed as single purpose, throw-away efforts. A model is built from scratch to address a particular problem or question, and then it is often discarded with little thought given to additional use. This single-use, throw-away mentality of modeling is very expensive, time consuming and wasteful. Second, lack of access to models by non-modeling specialists limits their usage and value. The availability of a modeling expert becomes a constraint on the usage and value-added contributions of the model.

The advent of the object oriented paradigm has provided an impetus for researchers to create modeling environments for building reusable simulation models (Bhuskute et al. 1992; Adiga 1989; Glassey and Adiga 1989). Designing for reusability involves identification of behaviors that are useful in more than one context. In general, this implies a system design which adheres rather strictly to the "one-component-one-function" doctrine (Glassey and Adiga 1989). If a component performs more than one function, its usage becomes limited to situations in which all of its functions are required. Alternatively, if a strict one-to-one functionality is maintained between component and function, the components truly become "building blocks" from which a total system model can be constructed (Pratt et al. 1994).

2 REUSABLE AND PLUGGABLE MODELING

Object oriented programming (OOP), a paradigm in which all program variables are represented as objects which communicate by means of message passing, is a significant advancement toward the development of multiple use, general purpose, plug-compatible models. OOP possesses four key concepts which facilitate this advancement: encapsulation, message passing, late binding, and inheritance (Budd 1991).

A key consequence of the reusability emphasis is the implementation of the separation concept (Pratt et al. 1994). The implementation of separation involves the creation of separate and distinct modeling primitives for physical elements, information flow, and control decisions. Traditional modeling approaches have not considered the separate and distinct modeling of physical, information, and control elements. For example, in many simulation languages, the constructs that are provided for information and control are frequently hard coded and dispersed into the model. This results in code that is hard to modify and difficult to use for multiple purposes.

Another advantage of the separation of physical, information, and control objects is that it allows the system modeler to think of these elements independently during model development. This provides a more natural modeling environment. In other words, when developing the physical model, the modeler need not be concerned with the information or control aspects. The process involves selecting the appropriate physical components without being constrained by concerns regarding how to model information flow. Similarly, information flow is considered without regard to physical objects. This independence facilitates the creation of models with a higher degree of integrity and greater flexibility relative to experimentation with the model.

Bhuskute et al. (1992) proposed a framework with reusability features for modeling and simulation of discrete part manufacturing systems. As suggested therein, a modeler can visualize each modeling object in terms of its physical, information, and control aspects. Visualizing physical and information components of a system as distinct elements is straightforward. In most cases, these elements are tangible and easily defined; for a manufacturing system, a lathe, a drill press, or an AGV is a physical component, whereas a bill of material or a routing sheet is an information component. Control components are potentially more difficult to grasp. When a control element interacts with the physical element it controls, it evaluates the state of the system on the basis of physical system status and other available information. Then an action is taken (i.e., a decision is made) based on an algorithm or a decision process. The decision is then communicated to physical, information or other control/decision components. The modeler can store not only the primitive objects but also the composite objects (i.e., a work station or an assembly department) in the object library, and couple them in a suitable manner to represent the real world manufacturing system.

This framework (Bhuskute et al. 1992) was initially designed for creating models of discrete part manufacturing systems which were to be exercised by a simulation tool only. A prototype modeling environment was developed in which the models created were highly reusable within a simulation context. From a broader perspective, the models created by this environment were not 'reusable' in the sense that they could not be exercised by any tool other than simulation. While this was a major limitation, the richness of the models created and the ease with which the models could be changed opened up new possibilities. The question was "Now that we have the ability to create and modify with ease detailed descriptions of complex manufacturing systems, how can we use these 'models' with other analysis tools?" Recognizing this opportunity, Duse et al. (1993) suggested that a modeling framework be designed where a model of the enterprise be created without keeping any tool or any specific problem as the context for model development. This led to the concept of a "base model."

3 BASE MODEL CONCEPT

A base model is an abstraction of a real world manufacturing system in the richest possible way (Duse et al. 1993). It acts as the substrate from which further abstractions can be made for providing tool-specific models. Thus the modeling activity is viewed as a separate and distinct activity from problem solving (which requires the use of specific tools and further abstractions). Thus, reusability with respect to tools is achieved through the concept of a "Base Model" and "Configurators/Translators" which configure/translate problem specific and tool specific models from the base model. More detailed discussion of *tool independent modeling* and the *base model* concept can be found in Kamath et al. (1995 and 1996). In the creation of a base model, a library of manufacturing modeling primitives, described earlier in this paper, is used. These primitives are classified into three types: physical primitives, information primitives, and control/decision primitives. A user can construct the manufacturing system specific base model by simply selecting appropriate modeling primitives and assembling them together using a software environment (Figure 1).

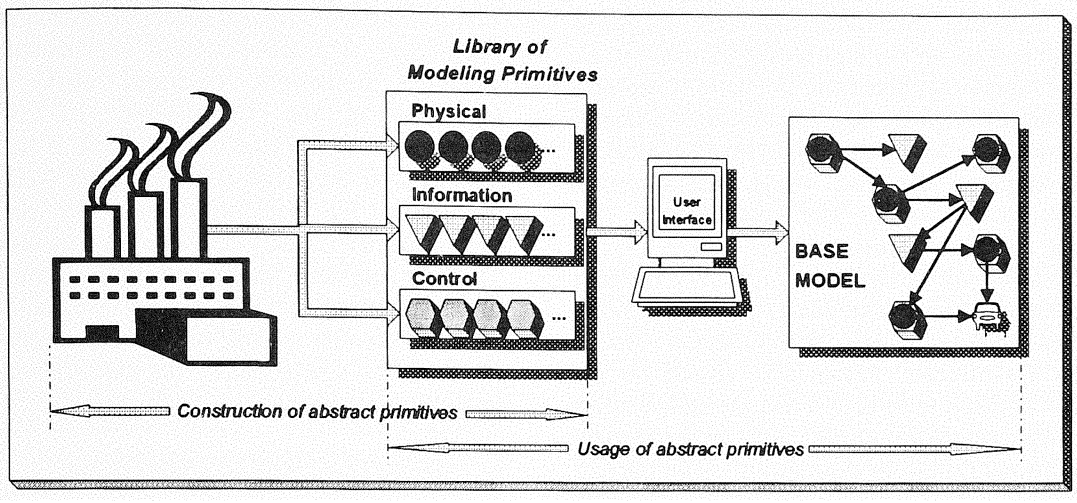

Figure 1: Construction and Usage of Abstract Modeling Primitives

A base model is never static. It evolves with the organization and is persistent in time. It is maintained as an ongoing activity just as company databases are created and updated on an ongoing basis. Maintaining the base model is thus a modeling activity for the sake of modeling and not pursued with an immediate specific purpose in mind, as is done in the traditional modeling approach which is purpose driven and tool specific. Given the existence of such a base model, one can derive tool-specific models, called execution models, to address a particular modeling problem or question at any time.

A comprehensive modeling environment should provide a variety of appropriate analysis, modeling, and optimization tools for solving problems which lie in different domains. For instance, rough-cut system design problems require aggregate estimates of performance measures and modelers usually employ analytical methodologies (e.g., queuing theory) for their solution (Suri and de Treville 1993). For determining detailed and accurate estimates of performance measures, modelers typically employ a simulation based approach. The base model, which is a tool independent representation of a specific manufacturing system, has all the information necessary to support the analysis tools available. Once an analysis tool is determined based on the specific experimental circumstances, the next step is to extract an execution model from the base model for the selected solver. Figure 2 illustrates this process.

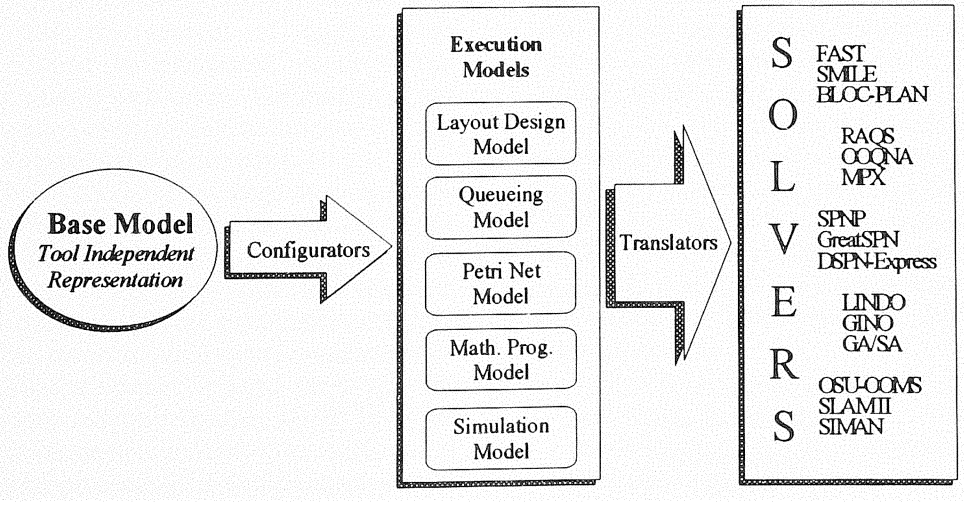

Figure 2: Tool Independent Model Representation

A new modeling framework has been developed based on the concept of the base model (Kamath et al. 1995). This framework, illustrated in Figure 3, includes both the construction phase and the utilization phase of the base model. The Manufacturing System (*1*) block in Figure 3 represents the real world manufacturing system. A Model Builder can construct the tool independent, generic, persistent Base Model (*3*) by using the Base Model Configurator (*2*). Once the base model is constructed a Decision Maker can seek answers to questions related to the manufacturing system through an Experimental Frame (*4*). Based on the nature of the problem to be solved and the complexities of the manufacturing system under study an appropriate Tool Specification (*5*) can be determined. With the tool specification in hand, a tool dependent configurator extracts the Execution Model (*6*) from the base model. Performing Analysis (*7*) will lead to either another experimental frame definition or Recommended Changes (*8*) for the system. As Changes are Implemented (*9*) within the manufacturing system, it is necessary to update the base model to reflect the changes.

The base model concept and modeling framework described in this section effectively address the first shortcoming of traditional modeling approaches, namely reusability. A proof-of-concept implementation of the modeling framework described above has been developed in the VisualWorks 2.0 development environment (ParcPlace 1994) which is based on the Smalltalk-80 language (Goldberg and Robson 1989). The user of the modeling environment can construct, save, and reuse a base model of a manufacturing system. The user can also analyze the manufacturing system performance using discrete-event simulation models, queueing network models, or Petri net models which are automatically configured by the modeling environment using the base model.

4 ORGANIZATIONAL CONTROLS

4.1 Reusable Controls

It is our experience that designing a framework to support the reuse of physical elements and information elements is relatively easy. Object oriented modeling environments have been developed by other researchers, examples of which are those developed by Glassey and Adiga (1989), Thomasma and Ulgen (1988), Narayanan et al. (1992), LeFrancois and Montreuil (1994). Most

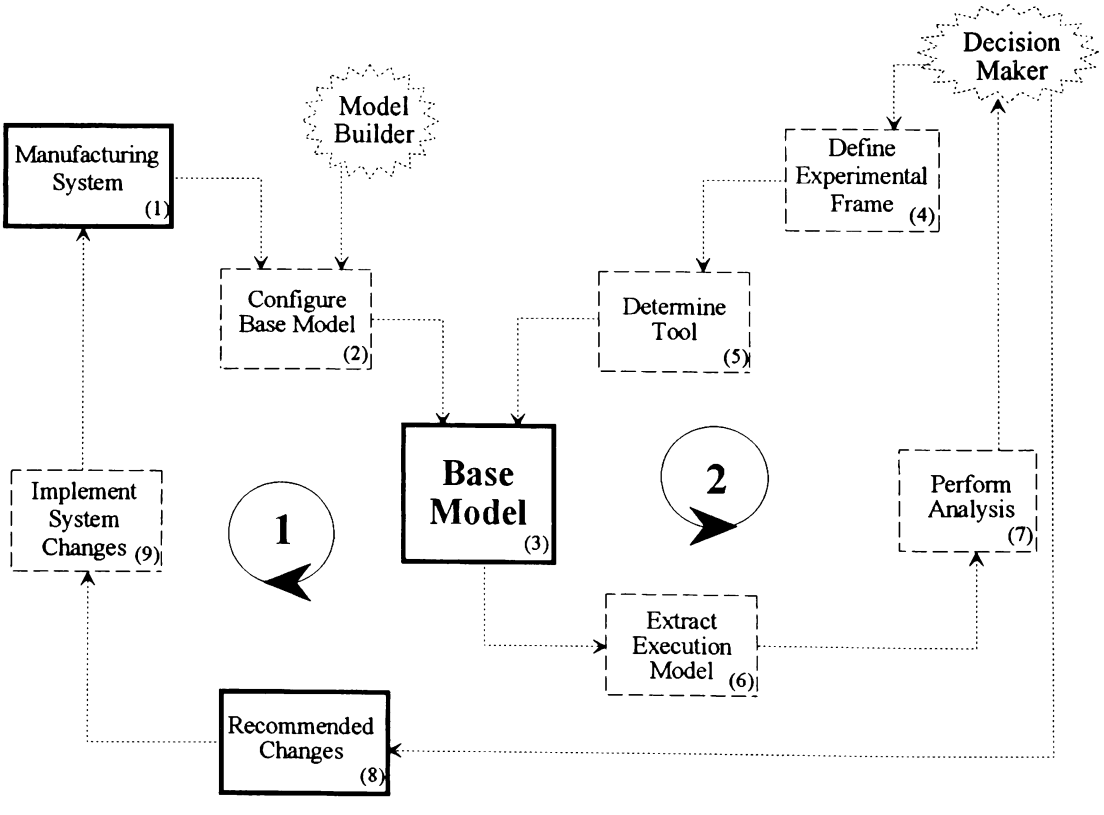

Figure 3: A New Framework for Manufacturing Systems Modeling

of the research efforts in the literature have been directed at the modeling of the physical elements and information elements. Furthermore, all of the efforts address simulation modeling only. The OOSIM group (Narayanan et al. 1992) and SmartSim/SmarterSim group (Thomasma and Ulgen 1988; Thomasma et al. 1990) have concentrated on lower level control for specifying material movements. Object oriented modeling and simulation of automated control in manufacturing has been proposed by Bodner et al. (1993). As per these authors, " ... the separation of control and plant resources in the model allows explicit representation of the factory control system, a critical feature of an automated system." However, in general, in the existing modeling environments, the control elements have been implicitly incorporated, and cannot be distinguished from other elements. That is, the control elements are interwoven with the other aspects of the model. This results in "forced hard coding" of control aspects. These modeling frameworks do not easily lend themselves to modeling organizational controls in a reusable way.

Achieving reusable organizational controls requires the modeler to consider the following two main issues:

1. *Identification of the decision points.* Decision points occur in three types of flows; the physical flow, the information flow, and the control flow. To identify the decision points, one must trace the threads of each of these flows. If one fails to recognize a decision point explicitly, then the logic for answering that decision-question can become hard coded, limiting the scope for achieving pluggable controls. The model developer must *be aware of default decisions.* These are the decisions that are not explicitly seen as decision making points, since a particular operating procedure is so ingrained that the concept of doing things differently cannot be recognized unless special attention is paid to such cases. For modeling, one needs to question every method to discover whether there exists a potential decision point.
2. *Identification of the set of questions that need to be answered at a decision point.* Granularity of the decision-question is a primary determinant of the degree of reusability that can be achieved. Composite questions (coarse granularity) will lead to logic that involves multi-dimensional decision making. Thus a change in logic along any one dimension may require re-coding of the complete logic. For example, consider the decision of customer order processing. The logic employed for answering this question may also involve determining the priority for the order if accepted, quantity of order accepted, and estimated delivery date based on which acceptance decision is made. While modeling, one needs to consider these aspects of the decision individually so that one does not group the logic for all these dimensions in one segment of code.

4.2 Pluggable Controls

Pluggability can be defined as a modeling capability that enables a modeler to replace a modeling object with a compatible one without changing the rest of the model. The modeling object can be a physical, information, or control/decision element. Since these elements can be grouped according to the levels in a manufacturing organization such as the machine level, the department level, and the plant level, we can potentially have pluggability at multiple levels. In our prototype implementation, we have achieved pluggability with respect to certain elements at different levels in the organization. Several characteristics of the OOP framework, such as polymorphism, inheritance, and encapsulation, simplify the implementation of pluggability.

Pluggability greatly facilitates the modeling of control logics in a manufacturing setting. The model user should be able to manipulate these control logics in order to find the 'optimum' set for a given physical and organizational configuration. Figure 4 presents a work station object for an illustration of pluggable control logics. The work station object has input and output buffers, a set of attributes (name, parent, location, footprint, status, reliability parameters) as well as a set of controller objects. The controller objects are of three functional classes; input queue controllers, output queue controllers, and resource controllers. Each functional controller class has a set of potential logics which can be used for answering a set of specific questions. For the input queue controller, these logics are grouped into four categories - part addition, part removal, primary part selection, and secondary part selection. In each of these categories there is a set of logics, as depicted in Figure 5, to answer the same question with different information usage. For each of the actions of part addition, part removal, and part selection, the model builder can substitute one logic element for another. In our prototype implementation, the user has the following choices.

- Add Checks (part addition) - *addAlways* (no restriction on adding a part), *capacity* (restriction on the queue size), *size* (restriction on the part size), *type* (restriction on the part type).
- Remove Checks (part removal) - *presence* (the only restriction is the presence of the part), and *productionOrderCompletion* (restriction is the

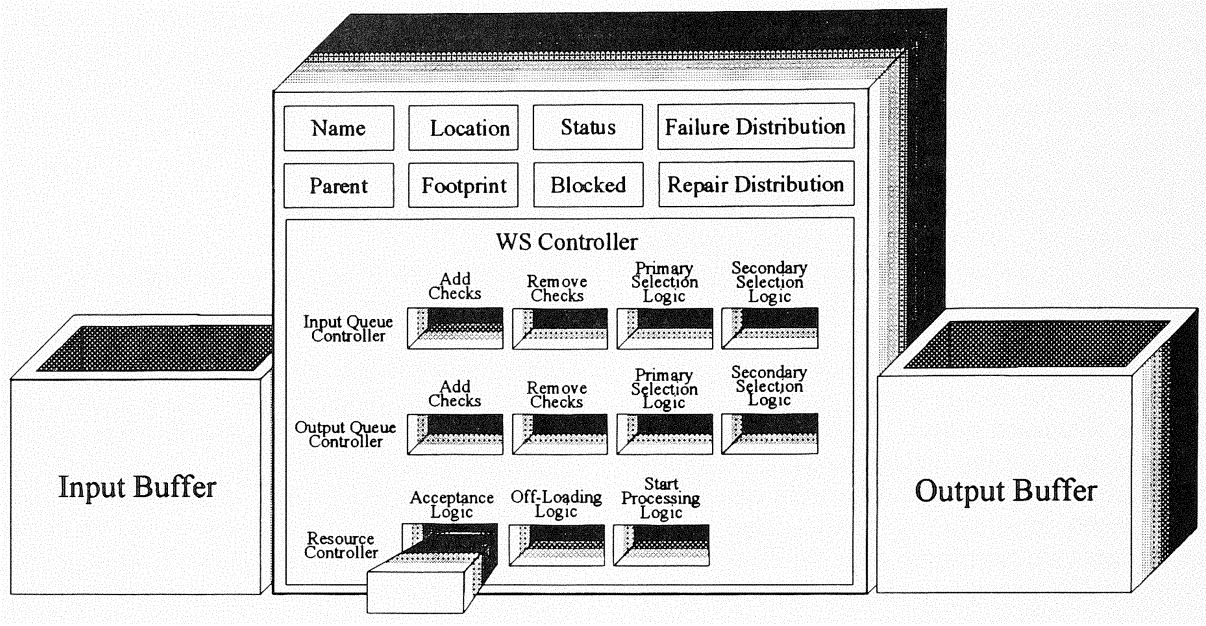

Figure 4: Pluggable Control Logics at the Workstation Level

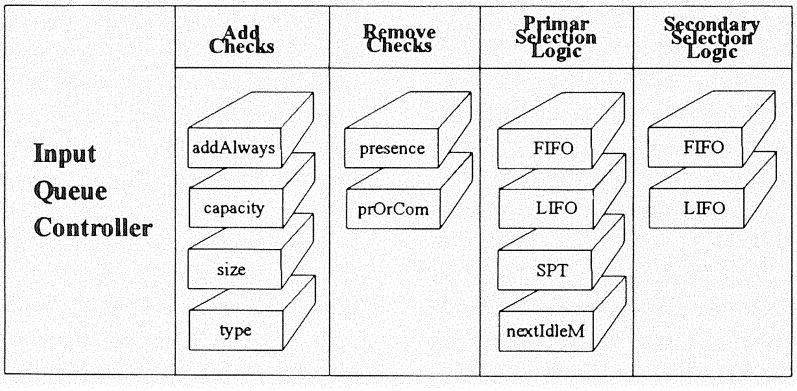

Figure 5: Alternative Logics for Input Queue Controller within a Workstation Controller

completion of the production order).

- Primary Selection Logic (part selection) - *FIFO* (first in first out), *LIFO* (last in first out), *SPT* (shortest processing time), and *nextIdleM* (select the part whose next processing machine is idle).
- Secondary Selection Logic (tie breaker) - *FIFO* (first in first out), and *LIFO* (last in first out).

In our control framework the internal implementation of a logic algorithm is transparent to the modeler so long as the controller can respond with appropriate feedback to a posed question. In order to answer the posed question, the logic algorithm might use complicated artificial intelligence techniques, simple calculations, or a set of if-then rules. Neither the user nor the model developer should be concerned about the internal structure of the control logics.

5 ONGOING RESEARCH

When this modeling environment is used for manufacturing system reconfiguration, the phases involved in responding to particular questions posed are often executed in an iterative fashion. Even the determination of the problem and the selection of the appropriate analysis/optimization tool requires extensive expertise. An intelligent advisor would be of great

An intelligent advisor would be of great benefit to non-modeling specialists (e.g., manufacturing managers) to determine a structured problem from a given set of symptoms and select an analysis/optimization tool to address such a problem (Delen et al. 1996). Furthermore, such an intelligent advisor can also help the user to analyze the output from several iterations and to determine if further investigation is needed. One of the main ongoing research efforts is directed at conceptualizing, designing, and implementing such an intelligent advisor.

ACKNOWLEDGMENTS

Support for this research is being provided by the National Science Foundation through Grant No. DMI-9400572. We also wish to acknowledge the support provided by the AT&T Foundation and the Oklahoma Center for the Advancement of Science and Technology (OCAST) during the formative period of this research. This work could not have progressed to its current level had it not been for the significant contributions of Prof. Joe Mize whose efforts are gratefully acknowledged. The efforts of former research assistants, especially Manoj Duse and Jagannath Gharpure, are also acknowledged.

REFERENCES

Adiga, S. 1989. Software Modelling of Manufacturing Systems: A Case for an Object-Oriented Programming Approach. *Annals of Operations Research* 17: 363-378.

Bhuskute, H. C., M. N. Duse, J. T. Gharpure, D. B. Pratt, M. Kamath, and J. H. Mize. 1992. Design and implementation of a highly reusable modeling and simulation framework for discrete part manufacturing systems. In *Proceedings of the 1992 Winter Simulation Conference*, ed. J. J. Swain, D. Goldsman, R. C. Crain, and J. R. Wilson, 680-688. Piscataway, New Jersey: Institute of Electrical and Electronics Engineering.

Bodner, D. A., S. D. Schneider, S. Narayanan, U. Sreekanth, T. Govindaraj, L. F. McGinnis, and C. M. Mitchell. 1993. Object-Oriented Modeling and Simulation of Automated Control in Manufacturing. In *Proceedings of the 1993 IEEE Conference on Robotics and Automation*, 3: 83-88. Atlanta, Georgia.

Budd, T. 1991. *An Introduction to Object-Oriented Programming*. Reading, Massachusetts: Addison-Wesley Publishing Company.

Delen, D., D. B. Pratt, and M. Kamath. 1996. A New Paradigm for Modeling of Complex Manufacturing Systems. In *Proceedings of The First Conference on Intelligent Manufacturing Systems*, ed. M. Ozbayrak, H. Taskin, E. Oztemel, 27-37. Sapanca-Sakarya, Turkey.

Duse, M. N., J. T. Gharpure, H. Bhuskute, M. Kamath, D. B. Pratt, and J. H. Mize. 1993. Tool-Independent Model Representation. In *Proceedings of the 2nd Industrial Engineering Research Conference*, 700-704. Norcross, GA.: Institute of Industrial Engineers.

Glassey, C. R., and S. Adiga. 1989. Conceptual design of a software object library for simulation of semiconductor manufacturing systems. *Journal of Object Oriented Programming* 2, 4: 39-43.

ElMaraghy, H. A. and T. Ravi. 1992. Modern Tools for the Design, Modeling, and Evaluation of Flexible Manufacturing Systems. *Robotics & Computer Integrated Manufacturing* 9, 4/5: 335-340.

Goldberg, A. and B. Robson. 1989. *Smalltalk-80: The Language*. Reading, MA.: Addison-Wesley.

Kamath, M., D. B. Pratt, and J. H. Mize. 1995. A Comprehensive Modeling and Analysis Environment for Manufacturing Systems. In *Proceedings of the 4th Industrial Engineering Research Conference*, 759-768 Norcross, GA.: Institute of Industrial Engineers.

Kamath, M., D. B. Pratt, D. Delen, S. Sivaramakrishnan, B. Krishnamoorthy, and R. Fernandes. 1996. An Advanced Modeling Environment to Support Rapid Analysis and Reconfiguration of Agile Manufacturing Systems. Submitted to *IIE Transactions*.

LeFrancois, P. and B. Montreuil. 1994. An Object-Oriented Knowledge Representation for Intelligent Control of Manufacturing Workstations. *IIE Transactions* 26, 1: 11-26.

Leung, Y. and R. Suri. 1990. Performance Evaluation of Discrete Manufacturing Systems. *IEEE Control Systems Magazine*, June, 77-86.

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, 3: 14-26.

Narayanan, S., D. A. Bodner, and C. M. Mitchell. 1992. Object-Oriented Simulation to Support Modeling and Control of Automated Manufacturing Systems. In *Proceedings of the 1992 Western Multi-Conference*, 1-15. San Diego, CA.: Society for Computer Simulation.

ParcPlace. 1994. *VisualWorks Release 2.0 User's Guide*. ParcPlace Systems, Inc., CA.

Pratt, D. B., P. Farrington, C. Basnet, H. Bhuskute, M. Kamath, and J. H. Mize. 1994. The Separation of Physical, Information, and Control Elements for

Facilitating Reusability in Simulation Modeling. *International Journal in Computer Simulation*, 4: 327-342.

Suri, R. and S. de Treville. 1993. Rapid Modeling: The Use of Queueing Models to Support Time-Based Competitive Manufacturing. *Operations Research in Production Planning and Control*, 21-30. Springer-Verlag.

Thomasma, T., and O. M. Ulgen. 1988. Hierarchical, Modular Simulation Modeling in Icon-Based Simulation Program Generators for Manufacturing. In *Proceedings of the 1988 Winter Simulation Conference*, ed. M. Abrams, P. Haigh, J. Comfort, 254-262. Piscataway, New Jersey: Institute of Electrical and Electronics Engineering.

Thomasma, T., Y. Mao, and O. M. Ulgen. 1990. Manufacturing Simulation in Smalltalk, *Object-Oriented Simulation*, ed. A. Guasch, 93-97. SCS.

AUTHOR BIOGRAPHIES

DURSUN DELEN is a Research Associate in the Center for Computer Integrated Manufacturing within the School of Industrial Engineering and Management at Oklahoma State University, Stillwater, OK. He received a B.S. degree in Industrial Engineering from the Istanbul Technical University, Istanbul, Turkey, in 1986, and an M.S. degree in Industrial Engineering from the Yildiz University, Istanbul, Turkey, in 1988. He is currently completing his Ph.D. degree in Industrial Engineering and Management at Oklahoma State University. He has three years of industrial experience in information systems analysis and design. His research interests include manufacturing systems modeling, discrete event simulation, object-oriented modeling, and expert systems. He is a member of Alpha Pi Mu, IIE, and INFORMS.

DAVID B. PRATT is an Assistant Professor in the School of Industrial Engineering and Management at Oklahoma State University, Stillwater, OK. He holds B.S., M.S., and Ph.D. degrees in Industrial Engineering from Oklahoma State University. He has twelve years experience in the petroleum, aerospace and pulp and paper industries. His research and teaching interests include manufacturing systems modeling, enterprise integration, and the strategic implications of CIM. He is a registered Professional Engineer in Oklahoma, a certified Fellow in Production and Inventory Management, and an ASQC Certified Quality Engineer. He is a member of IIE, NSPE, APICS, INFORMS, and ASQC.

MANJUNATH KAMATH is an Associate Professor in the School of Industrial Engineering and Management and Director of the Center for Computer Integrated Manufacturing at Oklahoma State University, Stillwater, OK. He received the B.Tech. Degree in Mechanical Engineering from the Indian Institute of Technology, Madras, India, in 1982, the M.E. degree in Automation from the Indian Institute of Science, Bangalore, India, in 1984, and the Ph.D. degree in Industrial Engineering from the University of Wisconsin-Madison, in 1989. His primary areas of interest are stochastic modeling and queueing theory, analytical performance modeling of manufacturing systems, object-oriented modeling and simulation of discrete-event systems and Petri nets. He is a member of IEEE, IIE, and INFORMS.