Proceedings of the 1992 Winter Simulation Conference
ed. J. J. Swain, D. Goldsman, R. C. Crain, and J. R. Wilson

SIMULATION OF ADVANCED MANUFACTURING SYSTEMS

Gerald W. Evans William E. Biles

Department of Industrial Engineering
University of Louisville
Louisville, KY 40292

ABSTRACT

This paper gives an overview of how simulation modeling techniques can be employed in the design and analysis of advanced manufacturing systems. The reasons for the complexities of these systems, as well as the uses of simulation software packages, are discussed. Finally, examples of simulation models of advanced manufacturing systems, as developed by the authors, are presented.

1 INTRODUCTION

Advanced manufacturing systems (AMSs) are those which manufacture parts with the material handling functions, machine operations, and machine tools under the control of a computer (Herald and Nof, 1978). The terms advanced manufacturing system, automated manufacturing system, and computerized manufacturing system have been used interchangeably in the literature (Gupta, 1990).

Examples of AMSs include flexible flow systems, flexible manufacturing systems, and flexible manufacturing cells. These systems can be composed of components such as robots, NC machining centers, automated guided vehicle systems, etc. The key characteristics of an AMS however include its flexibility (i.e., ability to produce a wide variety of parts with low set up times) and computer control.

The complexities of these systems basically result from their flexibility. These complexities make AMSs very difficult to design and operate. For example, Suri (1985) identified five phases of problems associated with the design and operation of flexible manufacturing systems: initial design; detailed design; installation; production planning, scheduling, and operation; and ongoing modifications. Each category of problems has its own particular set of design variables and performance measures; and, each set of decisions made with respect to one problem area affects the subsequent problem areas.

Examples of decisions which must be made in the design and operation of an AMS include the types of parts to be produced, the types and numbers of production machines and material handling equipment to include in the system, the layout of the system, the numbers of pallets/fixtures for each part type, the potential routings for each part type, local and global buffer capacities, lot sizes, sequencing rules, tool assignments, production rates, dispatching rules, etc. Some of these decisions (e.g., system layout) may be made only once every few years, while others may be made on a daily basis (e.g,. production schedules).

These decisions must be made while accounting for a variety of performance measures, including system cost and flexibility, due-date performance, quality of parts produced, production rates, inventory levels, machine utilizations, etc.

Simulation modeling can be a tremendous aid in the design process for an AMS. The purpose of this paper is to give the reader a brief overview of how this can be accomplished. Specifically, in the next section of the paper, we discuss the complexities associated with modeling AMSs. In the third section, we give a brief review of simulation software packages available for modeling AMSs. The fourth section of the paper contains a brief discussion of the steps of a simulation project, while the fifth section contains several examples of the modeling of AMSs performed by the authors. Finally, the last section of the paper contains a summary and conclusions.

2 COMPLEXITIES ASSOCIATED WITH THE MODELING OF AMSs

In basic terms, advanced manufacturing systems are complex because

1) A wide variety of parts are typically produced by an AMS, and
2) A number of different resources must interact in a complex fashion in order for the system to operate efficiently.

The system may be able to produce various part types simultaneously. Because of the typical AMS's flexibility, a particular type of part may follow any of several different routes. In addition to the complex routing decisions which may have to be modeled, decisions regarding the sequencing and scheduling of parts and resources must also be modeled. Again, this process is made more complex as a result of the variety of part types that can be produced by the system.

The types of resources associated with an AMS include pallets, fixtures, tools, robots, machines, conveyors, automated storage/retrieval systems, AGV's, AGV guidepaths, machine operators, maintenance personnel, inspection and testing equipment, etc. The simultaneous use of various resource types (e.g., a pallet, a machine, a conveyor, and a human operator) may have to be modeled. Hence, resource allocation decisions are not trivial.

Many researchers have recognized that the material handling subsystem of AMSs are typically very difficult to model (Chapter 13 of Law and Kelton, 1991). Again, this difficulty has to do with the complex interactions between and among material handling resources (e.g., AGV's, AGV guidepaths, conveyors, robots), production resources (e.g., machine tools), and parts.

Because of these modeling difficulties, some simulation languages have included special material handling modules. These modules can be viewed as simulators that can be employed within a larger system model constructed from the language. As an example, the SIMAN language allows an INTERSECTIONS element, a LINKS element, and a NETWORKS element, among other constructs for modeling AGV systems.

3 SIMULATION LANGUAGES AND SOFTWARE PACKAGES

A large number and wide variety of software packages are available to aid in the simulation modeling effort for manufacturing systems. In general, these packages correspond to either general purpose simulation languages, such as SLAM II, SIMAN, GPSS, and SIMSCRIPT II.5, or simulators such as WITNESS, ProModel, XCELL+, MAST, SIMFACTORY II.5, and AutoMod II. For comparisons/descriptions of these and other packages, see Law and Haider (1989), Chapters 3 and 13 of Law and Kelton (1991), Banks et al. (1991), and the Directory of Simulation Software (1991) from the Society for Computer Simulation. See the recent Proceedings of this conference for more in-depth descriptions of some of the packages (e.g., Goble (1991) for SIMFACTORY II.5, Harrel and Tumay (1991) for ProModel, etc.).

Banks et al. (1991) define a simulator as "a parameter-driven simulation that requires no programming." In general, a simulator can be thought of as being easier to use than a simulation language, but not as flexible. In recent years, the gap between simulators and simulation languages has narrowed. That is, many simulation language packages have been developed to the point where modeling constructs, which are in effect simulators, are included. For examples, see the material handling constructs of SIMAN discussed earlier (Chapter 9 of Pegden, Shannon, and Sadowski (1990)) and of SLAM II (Chapter 16 of Pritsker (1986)). On the other hand, simulators such as WITNESS allow programming inserts to increase the flexibility of their modeling capabilities.

In recent years, more attention has been paid to the use of simulation in the daily operation of an AMS. This is manifested, for example, in the development of the FACTOR/ AIM software package (Krahl, 1991) for scheduling manufacturing facilities in response to daily changes on the shop floor. The increased efficiency of hardware and software has allowed this type of development.

Another trend in recent years has been the development of packages to address all aspects of a simulation modeling project. A good example of this type of system in SLAMSYSTEM, a PC-based system which through the use of WINDOWS allows the user to manage files of various types (i.e., control files, network files, user insert files, output files) so that several versions of a model can be easily managed.

There is no <u>one</u> best package. That is different packages have different advantages/disadvantages when compared to other packages. For example, Banks (1991) lists 33 features to consider in determining which simulation software package to purchase. These features are categorized into input features, processing features, output features, environment features, and cost features. It is important that the buyer/user of the software have a clear understanding of what the requirements are for his situation. For examples, the software must be compatible with the available hardware, and the number and capabilities of the users of the software must be considered.

Many firms may find it desirable to purchase more than one package, for example, both a simulation language package for modeling very complex situations and a simulator for modeling less complex situations more quickly.

4 THE STEPS OF A SIMULATION PROJECT

Several authors have discussed the basic steps in all simulation studies. For examples, see Chapter 1 of Law and Kelton (1991), Chapter 1 of Pritsker

(1986), and Chapter 1 of Pegden, Shannon, and Sadowski (1990).

Law and McComas (1991) note that a simulation project is actually a sophisticated systems analysis activity. They describe the basic steps of a simulation project as follows:

1. Formulation (defining) of the problem and project planning.
2. Collection of the data and formulation of a model.
3. Validation of the model.
4. Coding and verification of a computer program.
5. Execution and analysis of the pilot runs.
6. Validation.
7. Design of experiments.
8. Execution of production runs.
9. Analysis of output data.
10. Documentation, presentation, and implementation of results.

Law and McComas also note that step 1 is often shortchanged; that is, "a careful statement of the problem's objectives is often neglected, due to a lack of understanding of the nature of simulation, the information it can provide, and the time and effort required for a sound study."

The main point of the Law and McComas article is that model coding typically represents only a very small part of a simulation project. Successful completion of simulation project requires attention to each of the steps noted above through the use of sound project management techniques. In the simulation modeling of AMSs, it's especially important to obtain a detailed description of the logic involved in the system operation. This is often difficult because of the complex nature of this system operation.

5 EXAMPLES

In this section of the paper, we detail several examples involving the modeling of an AMS. For these projects, the SLAM II language (as incorporated within the SLAMSYSTEM package) was used. The purposes of these examples are, first, to illustrate some of the complexities associated with the modeling of AMSs and, second, to illustrate how to model these complexities with a specific language/system. The SLAM II language was used, but these models could have been constructed using any of several different languages. In fact, alternative approaches even within the SLAM II language could have been used.

For other examples of simulation models of AMSs, see any of several textbooks on simulation modeling (e.g., Pegden, Shannon, and Sadowski, 1990) as well as the Proceedings of this conference (e.g., Davis, 1986 and Jeyabalan and Otto, 1991).

5.1 Modeling of a Semi-automated Assembly Line

This project involved the modeling and analysis of a semi-automated assembly line consisting of 29 stations. Because of the proprietary nature of the project, the specifics of the products produced by the line, and of the production operations performed, will not be disclosed here.

The purpose of the project was to develop a simulation model that would allow for experimentation with various design scenarios, involving changes to the current system, with respect to the:

1) number of back-up machines at each station,
2) locations and sizes of external buffers,
3) cycle times at the stations,
4) elimination of machine downtime,
5) velocity of pallets on the conveyor, and
6) number of pallets.

The product came in two different types of frames; the first frame type had four different types of models, while the second frame type came in five different types of models. Hence, nine different types of assemblies were produced by the system. The cycle times at the stations did not change from one type of assembly to another; however, there were set-up times involved at some stations when production was switched from one type of assembly to another.

The assemblies moved sequentially from one station to the next through the use of an accumulating (queueing) conveyor. Each assembly was contained on two different types of pallets, labeled A and B, as it moved through the system.

Most of the stations contained an "internal buffer" of size 1--i.e., one assembly at a time was processed at the station. However, a few of the stations contained an internal buffer with a capacity greater than one. For example, the sixth station was a washer that could accommodate 39 assemblies simultaneously.

The SLAMSYSTEM package was used to model the system. Each run of the model required the processing of three SLAMSYSTEM files: a control file, a network file, and a user insert file (consisting of FORTRAN code). Some 31 different types of resources were modeled -- one type for each of the 29 stations, and one type each for type A pallets and type B pallets.

Global variables, of type XX, were used to represent the type of assembly being produced by the system, the velocity of pallets on the line, and information about the production schedule. Global variables, of type ARRAY, were used to represent the cycle time, the changeover time (from one type of assembly to another), the external buffer capacity, and the type of assembly most recently processed for each station. Global (ARRAY) variables were also used to represent the distances between each pair of sequential stations or the conveyor.

5.1.1 Interfacing a Production Schedule With the Model

The production schedule was represented as a sequence of assembly types through the use of XX variables. For example, if the user wanted to process a run with 500 assemblies of type 1, 400 assemblies of type 4, and 1000 assemblies of type 3, then he would specify the following initial values for the XX variables:

XX(7) = 3, indicating 3 entries in the schedule,
XX(10) = 500,
XX (11) = 1,
XX (12) = 400,
XX (13) = 4,
XX (14) = 1000,
XX (15) = 3.

The global variable XX(8), initially set at a value of 1, was used to represent the current entry being processed in the schedule.

In this model, an entity (assembly kit) is created every 15 seconds by a CREATE node, but it will only be routed into the system if the number of assemblies waiting for pallets is less than 5. XX(9) is used as a counter to keep track of the number of assembly kits for a particular type of assembly; for example, when XX(9) attains a value of 501 for the first entry in the schedule, then XX(9) will be greater than XX(10) = 500, indicating that the next assembly to be processed should be the first of the 400 assemblies of type 4. XX(1) is used to keep track of the type of assembly currently being processed by the line; (i.e., XX(1) will assume the sequential values of 1,4, and 3 for this run). XX(8) is used to keep track of the number of the schedule entry being processed currently; (i.e., XX(8) will take on the sequence of values 1,2, and 3, for this run).

ATRIB(11), a local variable, is used as an indicator attribute for an assembly; that is, for all assemblies processed on the line except for the very last one (according to the schedule), ATRIB(11) will have a value of 0. The last assembly to be processed will have an ATRIB(11) value of 100. This information is used to stop the simulation run, when this particular entity (with an ATRIB(11) value of 100), reaches the end of the line. The model continues to create assembly kits though, even after this last kit, according to the input production schedule, has been created. The reason for this was to keep the station utilization unbiased.

5.1.2 Modeling of Internal Queues

Another relatively difficult aspect involved in modeling this system had to do with those stations with an internal buffer greater than one. These stations could not simply be modeled as queue nodes for the following reason. First, whether or not an assembly can enter the last position of the internal queue from the first position of the external queue depends upon whether there is an item in the last position of the internal queue. Hence, just because there is only one item in an internal queue with a capacity of three items, this does not mean that an item from the external queue can enter; that one item in the internal queue must also not be in the last position of the internal queue.

One possible way to model this situation could have been through the use of a service time based on a node release time. (see page 132 of Pritsker (1986)). To allow more generality, we took a slightly more complicated approach involving the use of AWAIT nodes, ENTER nodes, EVENT nodes, GOON nodes, conditional branching, and ALLOC subroutines. An AWAIT node is used to represent the external queue, and an ALLOC subroutine (a user-written FORTRAN subroutine) is called whenever an assembly arrives to the external queue, in order to check whether the assembly can enter the internal queue (according to the conditions discussed above).

Another user-written FORTRAN subroutine is called through the use of an EVENT node, and is executed immediately after an assembly exits the last position of an internal buffer. The code checks to see whether there are any assemblies waiting to enter the internal buffer from the external buffer. If so, the entity representing the assembly is removed from the file representing the external buffer and placed into the portion of the model representing the internal buffer through the use of an ENTER node.

Conditional branching from GOON nodes is used to check whether or not an item just finished with its operation in one position of the internal queue can advance to the next position. The "checking" is done at one second simulated time intervals; although this does not provide an exact representation of the programmable logic controller actually used in the system, it is close enough for modeling purposes.

5.1.3 Modeling the Blocking Aspects of the System

The assembly line was set up in such a way that whenever a station's external queue was full, the preceding station was blocked from sending an assembly to that next station. In effect, the preceding station was stopped from doing more work. This happened more often than one might expect, due to machine breakdowns and the varying cycle times at the various stations.

Often the easiest way of handling this situation in SLAM II is to use the blocking mechanism associated with QUEUE nodes. However, since we used RESOURCES and AWAIT nodes to model the stations, we employed a different approach involving the use of the STOPA activity duration.

As mentioned above, an AWAIT node is used to model the external buffer for a station. An EVENT node, which accesses a user-written FORTRAN subroutine, is used to set a value for ATRIB(10), a local variable. This ATRIB(10) value is used to represent the processing time at a station for an assembly, including the changeover time if the assembly being processed is a new one in the sequence, according to the production schedule.

Another EVENT node is used to check whether or not the external buffer for the next station is at capacity. If so, ATRIB(9), an indicator attribute for the entity, is set to a value of 1, and the assembly is sent along an activity with an "indefinite duration" of STOPA (ATRIB(3)), prior to releasing the resource corresponding to the current station. If the external buffer for the next station is not full, then ATRIB(9) for the entity is set to 0, the resource corresponding to the current station is released, and the entity is sent along an activity which represents the section of the conveyor connecting the current station to the next station. In addition, through the use of another EVENT node, any assemblies blocked at the previous station can be "unblocked", and therefore travel to the current station.

A variety of experiments were run with the model of the assembly line, including those related to varying the number of pallets in the system, varying the numbers of backup machines, varying the cycle times at the stations, etc. The plant engineers were intimately involved in the experimentation process, and hence had greater confidence in the model output.

5.2 Modeling a Semi-Automated Test/Repair Line

The completed assembly produced in the AMS described in section 5.1 then entered a test and repair line where it was tested at each of four stations and, if defects were found, repaired at yet a fifth station. The physical system and process flow presented a far less complex situation than did the assembly line, so that the test and repair line was entirely modeled using the network features of SLAM-II. The advantage of the network modeling approach was that the network flow almost exactly mimicked the digital control logic of the system.

In numerous instances in the test repair model, the simulation model behaved exactly like the programmable logical controller (PLC) which monitored and controlled that segment of the system. For example, an entity arriving at a GOON node represented a pallet arriving at a processing station and tripping a switch. The PLC logic was as follows:

i. If the processing station (an AWAIT node representing that resource) was open (FREE), the test pallet (entity) containing the assembly was placed in the processing station and the operation (service activity) commenced;

ii. If the processing station was busy, an activity was initiated which rerouted control back to the GOON node with a delay of 1 second, just as the PLC would do. This "logic loop" was repeated every 1 second until the processing station (AWAIT node) was available (FREE).

The ultimate advantage of the test/repair line simulation model was that the manufacturing engineer in charge of the line could evaluate changes in equipment layout as well as PLC control logic in a matter of minutes. For example, a change in logic sequence was as simple as inverting two statements in the SLAM-II network model, while relocating the repair station away from the other test equipment involved nothing more than changing the destination (LABEL) of an entity in an ACTIVITY statement. Adding a new machine to the line was somewhat more involved, and required as much as 10 minutes to effect the required alteration to the SLAM-II simulation model.

6 SUMMARY AND CONCLUSIONS

Needless to say, having a valid simulation model available gave the manufacturing engineer in the above examples much greater confidence in making alterations in the existing production system. For just a few thousand dollars, the manufacturing engineer was able to purchase a PC version of SLAMSYSTEM, engage consultants to show him how to model the production system for which he was responsible, and evaluate investments in new production equipment costing in excess of $1 million. He was then able to show his plant manager just how to achieve the corporate production goals for a new product line.

The key to this manufacturing engineer's success was that, through simulation, he was able to develop very detailed and accurate models of the AMS. He saw the need to have a tool that allowed him to evaluate changes in material flow, equipment placement, operating policies, and PLC logic before experimenting with those changes in the actual system. The fact that the model predicted the effect of proposed line changes within 2 percent gave him the confidence to employ simulation for more involved and expensive alterations in production system design.

REFERENCES

Banks, J. 1991. Selecting simulation software. *Proceedings of the 1991 Winter Simulation Conference*, eds., B.L. Nelson, W.D. Kelton, and G.M. Clark, **15-20**, The Society for Computer Simulation, San Diego, CA.

Banks, J., E. Aviles, J.R. McLaughlin, and R.C. Yuan. 1991. The simulator: new member of the simulation family. *Interfaces*. **21**, 76-86.

Davis, D.A. 1986. Modeling AGV systems. *Proceedings of the 1986 Winter Simulation Conference*, eds., J. Wilson, J. Henriksen, and S. Roberts, 568-574, The Society for Computer Simulation, San Diego, CA.

Directory of Simulation Software. 1991. Elliot Estrine (Ed.), Society for Computer Simulation, **2**.

Goble, J. 1991. Introduction to SIMFACTORY II.5. *Proceedings of the 1991 Winter Simulation Conference*, eds., B.L. Nelson, W.D. Kelton, and G.M. Clark, 77-80, The Society for Computer Simulation, San Diego, CA.

Gupta, M. 1990. An evaluation of operations planning and scheduling problem in advanced manufacturing systems. Ph.D. Dissertation, Department of Industrial Engineering, University of Louisville, Louisville, KY.

Harrel, C.R., and K. Tumay. 1991. ProModel tutorial. *Proceedings of the 1991 Winter Simulation Conference*, eds., B.L. Nelson, W.D. Kelton, and G.M. Clark, 101-105, The Society for Computer Simulation, San Diego, CA.

Herald, M.J. and S.Y. Nof. 1978. The optimal planning of computerized manufacturing system. Report No. 11, School of Industrial Engineering, Purdue University.

Jeyabalan, V.J., and N.C. Otto. 1991. Simulation models of material delivery system. *Proceedings of the 1991 Winter Simulation Conference*, eds., B.L. Nelson, W.D. Kelton, and G.M. Clark, 356-364, The Society for Computer Simulation, San Diego, CA.

Krahl, D. 1991. Tutorial: scheduling manufacturing systems with FACTOR. *Proceedings of the 1991 Winter Simulation Conference*, eds., B.L. Nelson, W.D. Kelton, and G.M. Clark, 128-131, The Society for Computer Simulation, San Diego, CA.

Law, A.M., and S.W. Haider. 1989. Selecting simulation software for manufacturing applications. *Proceedings of the 1989 Winter Simulation Conference*, eds., E.A. MacNair, K.J. Musselman, P. Heidelberger, 29-31, The Society for Computer Simulation, San Diego, CA.

Law, A.M. and W.D. Kelton. 1991. *Simulation Modeling and Analysis*, Second Edition. New York: McGraw-Hill.

Law, A.M. and M.G. McComas. 1991. Secrets of successful simulation studies. *Proceedings of the 1991 Winter Simulation Conference*, eds., B.L. Nelson, W.D. Kelton, and G.M. Clark, 21-27, The Society for Computer Simulation San Diego, CA.

Pegden, C.D., R.E. Shannon, and R.P. Sadowski. 1990. *Introduction to simulation using SIMAN*. Highstown, New Jersey: McGraw-Hill, Inc.

Pritsker, A.A.B. 1986. *Introduction to Simulation and SLAM II*, Third Edition. New York: John Wiley and Sons.

Suri, R. 1985. An overview of evaluative models for flexible manufacturing systems. *Annals of Operations Research*. **3**, 13-21.

AUTHOR BIOGRAPHIES

GERALD W. EVANS is an associate professor in the Department of Industrial Engineering at the University of Louisville. He received a B.S. in Mathematics in 1972, an M.S. in Industrial Engineering in 1974 and a Ph.D. in Industrial Engineering in 1979, all from Purdue University. Prior to his current position, he was an assistant professor in the School of Business at the University of Louisville (1981-1983), a senior research engineer at General Motors Research Laboratories (1978-1981), and an industrial engineer for Rock Island Arsenal (1974-1975). He is an associate editor for **IIE Transactions** and has co-edited the book, **Applications of Fuzzy Set Methodologies in Industrial Engineering**. His research interests includes multicriteria optimization and simulation modeling, especially as applied to problems in manufacturing system design and operation, engineering management,

and the service industries. He is an active member of IIE, TIMS, ORSA, and DSI.

WILLIAM E. BILES is the Edward R. Clark Professor of Computer-Aided Engineering in the Department of Industrial Engineering of the University of Louisville in Louisville, KY. He is currently involved in research in two principle areas: (1) simulation modeling of automated manufacturing systems, and (2) computer-integrated manufacturing of plastics. He is editor for simulation of two journals: **IIE Transactions,** and **Computers and Industrial Engineering**. Dr. Biles received the BS in Chemical Engineering from Auburn, the MSIE from the University of Alabama-Huntsville, and the Ph.D. in IEOR from Virginia Polytechnic Institute and State University. He has held industrial positions with Union Carbide and Morton Thiokol, and faculty positions at Notre Dame, Penn State, and LSU. He is a Fellow of IIE and chairman of the GRE engineering exam committee for Educational Testing Service.