Proceedings of the 1989 Winter Simulation Conference
E.A. MacNair, K.J. Musselman, P. Heidelberger (eds.)

INTEGRATING IBIS SIMULATIONS AND SYSTEMS PLANNING MODELS THROUGH MULTIPLE MODEL COMMUNICATIONS

Michael G. Ketcham
Ramesh Rajagopalan
Department of Industrial Engineering & Operations Research
University of Massachusetts
Amherst, Massachusetts 01003

ABSTRACT

This paper describes a multiple-process, multiple-windowed environment that allows users to interact with several models concurrently, including IBIS simulations. One goal in designing this environment is the transparent integration of simulation with other types of models, such as mathematical models for production planning. Its capabilities include interactive development of experiment specifications, automatic reconfiguration of simulations based on changing system specifications, and concurrent execution of simulations and production scheduling models to provide a detailed analysis of system capacities. Concurrent execution is controlled through techniques for multiple model communication.

1. INTRODUCTION

One goal in designing simulation environments is the transparent integration of simulation with other types of models, such as mathematical models and models based on artificial intelligence techniques. Transparent integration implies several capabilities. These include sharing information across multiple models, managing user interaction, and automatic reconfiguration of simulations based on changing system specifications. Developing such an environment requires (1) developing an information model and modeling database used to hold system descriptions, (2) developing simulation routines that are driven directly off of data specifications, and (3) developing a software architecture for multiple model communication (MMC).

This paper emphasizes MMC, although it touches briefly on other issues in simulation system design. It describes a multiple-process, multiple-windowed environment based on IBIS simulations, implemented on a workstation-class computer. This environment allows users to interact with several models concurrently. User interactions include the interactive construction of simulation models guided by the structure of the information model, interactive development of experiment specifications, and concurrent execution of simulations and production scheduling models to provide a detailed analysis of system capacities.

Among the most important motivations for developing multiple-model environments is the rapid evolution of both manufacturing and in modeling techniques. Particularly in multiple-plant, multiple-stage production companies, manufacturing planning involves complex planning hierarchies. These include corporate, plant, department, and shop floor planning. Yet we do not know in advance or from system to system what the structure of that hierarchy will be or what the planning requirements will be at a given level of the hierarchy. A multiple model environment will allow analysts to move between planning levels to appropriate planning strategies at each level.

Just as we do not know in advance what future planning hierarchies will be, we do not know what manufacturing technologies will be developed or what planning techniques will be used to represent those technologies. Rather than trying to anticipate planning requirements in a single model or a rigidly prescribed set of models, we expect to see flexible planning environments actually comprised of multiple models representing different facets of a manufacturing enterprise. A multiple-model environment allows users to select of models with different solution characteristics, range of applicability, solution speed and accuracy, or measures of performance. Similarly, it allows a single solution methodology, such as simulation modeling, to be brought to bear against different problems at different organizational levels in the planning hierarchy. We have focused on simulation, in particular, since simulations tend to be "closed boxes," in most application areas, which cannot be integrated with other types of models without extensive special purpose programming.

By pursuing a multiple-model environment we have generated a database-centered modeling environment for integrating computational and simulation models of manufacturing systems. By database-centered environment we mean development of a production database that holds facilities configurations, production requirements, and manufacturing parameters that can be accessed and updated by several planning and control models. A database-centered design allows communication between models that may have different data requirements or be written in different languages. In this environment, the database controls MMC, and preserves a high degree of data independence so that the database schema can evolve and intelligent interfaces be developed without needing to change models that have already been implemented. Because the data model exists independently of any one planning model, modeling techniques can evolve independently of other models without affecting the common view of the system shared between them.

2. MULTIPLE MODEL COMMUNICATIONS

Before going further in explaining MMC we need to define a number of terms related to modeling and model communication.

First, the word "model" itself is used in three contexts in this paper.

1. A **system model** is used to refer to a set of data whose elements represent the variables and their relationship in a real system. It provides information about the components and the dynamics of the real system for which the planning will be made.

2. A **data model** specifies objects, attributes, and information relationships needed to implement a particular system model. More simply, the data model allows us to create the database containing information about the system.

3. A **planning model** is a technique by which a particular planning problem can be solved to a particular degree of accuracy. It accesses the system model for the necessary information about the real system and converts the problem in to a suitable and solvable form.

In their discussion of decision support systems, Sprague and Carlson (1982) argue that one important reason for the disuse and misuse of planning models is the lack of multiple model communication in the decision making process. The models are used, in most cases, as stand-alone models and the model communication is left to the decision-maker as a mental or manual process.

Rather than standing independently, however, modeling results may in fact have complex interrelationships. In one case, planning models may address different facets of a planning problem. For example a demand forecasting model and master scheduling model use different techniques to address two aspects of long-term production planning. In a second case, two or more models may address the same facet of the planning problem. For example a scheduling model and dispatching model may both be used to generate job release decisions, or a scheduling model and simulation model may both be used to develop and evaluate production schedules.

In both these cases we can expect some degree of dependency among planning models. In the first group, the output from the demand forecasting model is an input to the master scheduling model; in the second group, the output from the scheduling model can be fed to a simulation model to evaluate the job release schedule.

More specifically, models can communicate with four degrees of interaction that we have called independent, sequential, iterative, and concurrent execution.

In **independent modeling** two or more planning models read a system description from the same database and execute independently.

In **sequential modeling**, results from one planning model are passed as inputs to a second model.

In **iterative modeling**, results from one model are returned as feedback to a second model to correct modeling errors or to update state variables based on additional analysis.

Concurrent modeling is the most complex form of interaction. In this case, two models are run simultaneously with shared results posted between them by way of a shared database.

The transfer of information between models can use **one-way** or **two-way** communication. One-way communication is comparable to sequential modeling, where results of one model are passed to a second. Commonly, the first model runs completely and posts its results only after it finishes execution. Two-way communication is comparable to concurrent modeling. Here there is a dynamic synchronization of information wherein all the participating models are concurrently run, and the information is transferred "to and fro" between the models.

Kantowitz and Sorkin (1983) describe the communication process as consisting of three blocks between the source and the destination: encoder, decoder, and channel. The encoder encodes the information from the source; the channel routes the encoded message to the decoder; the decoder decodes the information, and the decoded message reaches the destination. Thus, in one-way communication, the encoder block is at the source, the decoder is at the destination, and the channel allows information in only one direction. In two-way communication, an encoder and decoder are both available at the source and the destination, and the channel is bidirectional since, in two-way communication, the destination is also a source, and the source is also a destination.

An important assumption in this sequence of events is that the source and the destination are appropriately prepared so that correct information is available at the source, and the destination is ready to handle the communicated information. For this assumption to hold, some kind of initial processing has to take place at the source and the destination points, so the initial three-part communication model can be enhanced by adding an **initializer** which will prepare the source and destination.

We have used this theoretical configuration to design an MMC architecture in which the planning models are at the source and the destination, the database becomes the channel, and data extractors associated with each model behave as encoder, decoder, and initializer (Figure 1). These communications are implemented using the UNIX socket mechanism for data transfer between models, where a **socket** is an addressable endpoint for communication between two programs. Most applications of sockets use a "client" and "server" communications model. A Client program initiates a connection and requests service. A server program waits and accepts or rejects the connection. Once the Server and the Client form a connection, either program can send or receive data or terminate the link at any time (see Rochkind, 1985). In our implementation, the client is a supervisory program called the **model manager**. The planning models serve as multiple clients. In a windowing environment, each process can execute in its own window and maintain its own user interactions, while information is transferred between concurrently executing programs along data sockets.

Finally, MMC needs to manage control communications as well as data transfer. Control communications include establishing and breaking communication linkages between different components of the system, scheduling and synchronizing model execution, handling various types of signals so that an experiment can run smoothly, and managing user interactions. Control communications are managed by the model manager. The interface to the model manager allows the user to select experiments and planning models based on the planning requirements for a particular session, and to choose a manual or automated model of operation. In the manual mode, the user can communicate with

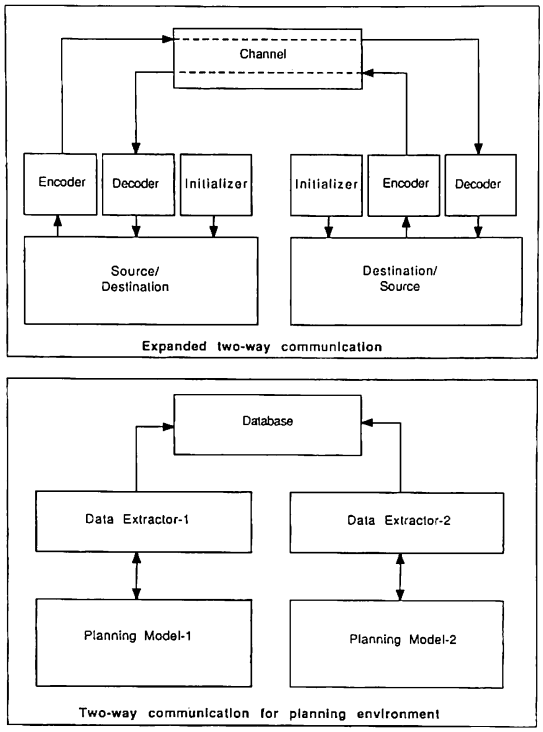


Figure 1. Expanded two-way communication model and implementation for MMC

planning models as they execute. In automatic mode, planning models transfer data automatically, without user intervention.

3. MODEL INTERACTION

Figure 2 shows the overall structure of a multiple model planning environment. Currently, our environment combines a multi-level scheduling heuristic and a simulation modeling component that automatically configures models based on production specifications held in a database. Both the scheduling and simulation models can represent complex assembly relationships with multiple resource requirements, multi-level bill-of-materials relationships, and sequencing constraints. In concurrent execution, they communicate through a common database so that the simulation is driven by the scheduler and the scheduler is updated in response to simulated system status regarding resource availability, job releases, and job completions.

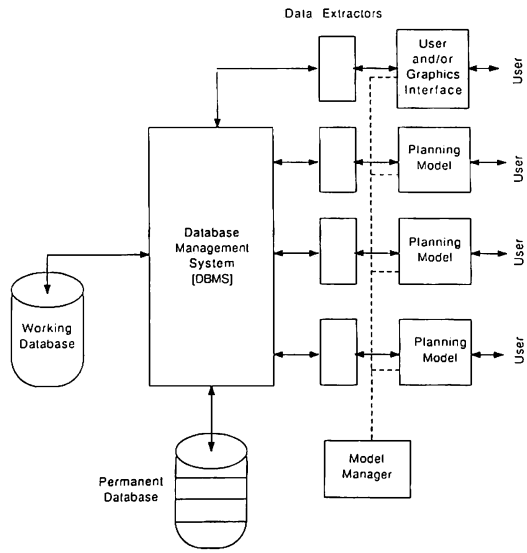
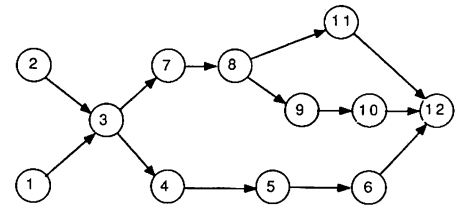


Figure 2. System architecture for MMC

MMC between the scheduler and simulation has been tested extensively against a set of assembly problems with a moderate variety of product types, but possibly complex assembly networks associated with each product. Figure 3 shows the production sequence for a representative product type. As this figure shows, each product may require one or more subassemblies to be completed before operations can take place, with precedence relationships among operations. There are also many-to-many relationships between operations and the resource types. A resource can be used in more than one operation and operations may require more than one type of resource, all of which must be available before the operation can begin so that different products and operations will compete for resources.



Node	Operation	SubJob	Res.Consumption		
			1	2	3
1	OPR1	SUBASSY2	2	0	0
2	OPR2	SUBASSY1	4	0	0
3	OPR3	SUBASSY1	0	2	0
4	OPR4	BYPROD1	3	0	0
5	OPR5	BYPROD1	2	0	3
6	OPR6	BYPROD1	0	0	2
7	OPR7	BYPROD2	0	0	3
8	OPR8	BYPROD2	2	0	2
9	OPR9	BYPROD2	5	3	2
10	OPR10	BYPROD2	2	2	2
11	OPR11	BYPROD3	0	2	0
12	OPR12	BYPROD3	2	0	0

Figure 3. Representative production requirements

Figure 4 shows a portion of the database files that represent these assembly relationships and that are used directly to drive the simulated system.

Test problems have included five different product types, which can be run simultaneously in the system using three types of resources. There will be more than one unit of each resource type available, but resource availability may vary over the planning horizon.

>File: OpTime

Operation	Entity	ProcTime
Opr8	ByProd2	NORMAL(2.3,.2)
Opr9	ByProd2	NORMAL(5.3,.5)
Opr10	ByProd2	NORMAL(1.1,.1)
Opr11	ByProd3	NORMAL(12.3,1.2)
Opr12	ByProd3	NORMAL(2.1,.2)

>File: OpRequirements

Operation	Entity	ReqEntity	NumInput	NumOutput	SourceQue
Opr8	ByProd2	ByProd3	0	1	-
Opr8	ByProd2	Resource-1	2	2	-
Opr8	ByProd2	Resource-3	2	2	-
Opr9	ByProd2	Resource-1	5	5	-
Opr9	ByProd2	Resource-2	3	3	-
Opr9	ByProd2	Resource-3	2	2	-
Opr10	ByProd2	Resource-1	2	2	-
Opr10	ByProd2	Resource-2	2	2	-
Opr10	ByProd2	Resource-3	2	2	-
Opr11	ByProd3	Resource-2	2	2	-
Opr12	ByProd3	ByProd1	1	0	ByProdQue
Opr12	ByProd3	ByProd2	1	0	ByProdQue
Opr12	ByProd3	Resource-1	2	2	-

Figure 4. Database representation of assembly relationships

Processing times are probabilistic, although the scheduling algorithm treats them as deterministic and integer. In addition to processing times, other probabilistic elements affect system performance. System resources may fail with probabilistic times between failures and down-times. New jobs arrive at the system with probabilistic inter-arrival rates, and preemption may be allowed for jobs in process if preemption is necessary to readjust the production schedule.

3.1. Scheduling Component

The scheduling component in our current environment has been adapted from a scheduling heuristic developed by Norbis and Smith (1986) for resource- constrained scheduling problems. This has proven to be a flexible and powerful schedule generator than that can handle multiple products with distinct process flows and competing resource requirements; several types of dynamic events; and hierarchical planning based on batching of production lots.

In its simplest outlines, the scheduling module produces a schedule in three steps:

(1) It divides operations into three priority sets based on a critical path methodology.

(2) The scheduler schedules operations in each priority set in order to maximize utilization of critical resources.

(3) It schedules remaining operations (those that are not on any critical path) where feasible.

The scheduler reads product specifications from a specially formatted numerical data file which is compact but which is difficult to generate or interpret. As a result, one task of the data extractor associated with the scheduler is to interpret database entries describing product specifications and translate them into the input format required by the scheduler. A second task is to translate the schedules and error messages generated by the scheduler from its internal notation back to a shared database format that can be accessed by the simulation.

The scheduler also responds to four types of events which can be triggered by the executing simulation and must be communicated to the scheduler in the appropriate format. These are: (1) new job arrivals; (2) deviations in observed processing times from expected processing times; (3) changes in resource availability; and (4) changes in job due dates.

3.2. Simulation Component

Earlier research by Ketcham has examined information requirements for manufacturing simulation (Ketcham, 1988; Ketcham, Shannon, and Hogg, 1989). The software developed as part of this research is called IBIS, which stands for "Information-Based Integrated Simulation," and constitutes a general-purpose simulation environment. IBIS modeling techniques allow for the generic description of a large class of manufacturing systems and information network, including complex bills-of-materials and multi-resource manufacturing operations, flexible production scheduling, and the representation of both "push" and "pull" inventory control.

More formally, IBIS is a database-centered simulation environment, where objects, relationships, and attributes in a simulated system are represented by database entries. The IBIS database stores specifications about a system to be simulated. IBIS simulations then read system specifications directly from the database without needing to generate an intervening model. As a result, a model does not exist as programming code but exists in the form of information stored in the database that is selectively retrieved in executing different models and different experiments. Because IBIS uses a database rather than programming code to control simulation execution, simulation values can be easily accessed and modified as a simulation runs by using database management commands. This capability allows users to develop automatically reconfigurable simulations based on database entries and has simplified the development of data extractors for concurrent model execution.

3.3. Model Management

Each experiment has a defined scope and can be associated with one or more planning models. When a user selects an experiment, a subset of the main (permanent) database is copied into a working database, based on the scope and other specifications for the selected experiment.

The modeling database is designed in such a manner that a user can select from several experiments and planning models to evaluate the control policies or to project system performance. Experiments specifications can control the characteristics of the system being modeled, and identify the planning models to be executed as part of the experiment. Different experiments can, for example, execute over different planning horizons, identify jobs to be considered in a particular planning problem, add new products or remove products from the experiment, or change input rates or number of pieces in process at the start of the simulation.

When a user begins a session, the model manager opens a port to the database so that the user can interactively select a suitable combination of an experiments and planning models. It also provides a view of the selected portion of the database to assist the user in specifying experimental details. Based on experiment specifications, a data extraction module searches the database to retrieve the required information about the system and extract a subset of the system model.

Depending on the experiment specifications, the model manager determines which planning models will be executed: For example, if the user selects an experiment which uses a simulation model independently, it extracts the modeling information needed for this one experiment. Since the data requirements of IBIS simulations match the way data is stored in the database, simulations are loaded without a dedicated data extractor. For other planning models, the model manager formats data as required for each planning model through dedicated data extractors. The model manager opens a window for each planning model and immediately sets up the control communications to synchronize information flow between models.

When running concurrently, the simulation is driven by the scheduler and the scheduler is updated in response to simulated system status in a repeated cycle: (1) The scheduling model is initialized and an initial schedule is obtained. (2) The database is updated with schedule information. (3) The simulation reads this schedule and schedules entities on the event calendar according to the posted schedule. (4) Events produced by the simulation that move the system off schedule are posted to the scheduling model, which recalculates the schedule and updates reorder information the database.

Changes in the simulated system, such as changes in resource availability, new job arrivals, changes in job due-dates, or variations in processing times, are stored in the database. At each time unit, the data extractor associated with the scheduling model accesses the database for event information which will be posted to the scheduler prior to recalculating a schedule, if necessary. Once the database is updated, the simulation is started, and is continued until an event occurs. In our implemented version, the model manager has included additional operations so that user interaction is not needed. That is, when the model manager detects an "EVENT" message, it immediately reads the event information and automatically posts the event information to the scheduling model, stopping the simulation model until a new schedule is calculated. When a new schedule is obtained from the scheduling model, the model manager interprets the schedule and posts the job release schedule to the database.

We have run five planning strategies against each of two problems These strategies are:

1. Apply simulation model alone with FIFO scheduling.

2. Apply scheduling model alone.

3. Apply simulation model driven by the initial schedule obtained from the scheduler. Schedules are recalculated whenever a new job arrives, but are not updated in response to machine failures, increases or delays in processing, or other events.

4. Apply simulation model with the schedule obtained dynamically from the scheduling model, with preemption of operations not allowed. All four types of events are allowed.

5. Apply simulation model with the schedule obtained dynamically from the scheduling model, with preemption of operations allowed. All four types of events are allowed.

These planning strategies are sufficient to test the MMC architecture. Strategies 1 and 2 require application of a single planning model whereas strategies 3, 4, and 5 require application of two planning models, with the strategies 4, and 5 needing dynamic synchronization of the results between the two planning models. These strategies, thus, provide a way to test MMC for independent, sequential, and concurrent execution of planning models.

Several experiments were run using each planning strategy. For example, initial experiements with the simulation run independently indicate that resource levels can be reduced to 9, 4, and 3, with new jobs of all five times arriving (approximately) every 100 time units. Any further reduction the resource levels would result in an infeasibility.

For strategy 2, in which the scheduling model is run independently, the results were obtained only for the initial set of jobs. Since the simulation model was not coupled to the scheduling model, the scheduler had no information about new job arrivals coming after the initial schedule was computed. Because of this restriction, the results obtained by running the scheduling model independently cannot not be compared directly with the results obtained by adopting other strategies. However, running the scheduler and simulation iteratively (Strategy 3), and running the scheduler and simulation concurrently (Strategy 4 and Stategy 5) gives insight into how the scheduler's assumption of deterministic processing times and the scheduler's priority sets perform in probabilitistic conditions.

Although results need to be tested further, these trial problems suggest that preemption is not a good strategy for this system, and that elaborate scheduling rules are not significantly better than a FIFO rule when resource utilizations are less than approximately 50%. Dynamically updating the schedule in response to events does provide some improvement in overall system performance by increasing resource utilization and maximizing overall earliness, although this improvement is marginal and may not be justified if running the scheduler is computationally expensive. In this case, the best policy would appear to be running from the initial schedule until new jobs arrived in the system.

4. CONCLUSION

From the point of view of MMC, these results show that simulation can be "opened up" to interact transparently with computational models by way of a shared data model and interprocess communications taking place through special-purpose data extractors. The concurrent sessions with the simulation and the scheduling models provide a testbed for evaluating deterministic models in a probabilistic environment, and they have indicated that a multiple model environment would allow for powerful capabilities in sensitivity analysis, error detection, and error recovery. There will be errors in any model due to approximations and simplifications in numerical formulas. In many cases, these errors can be compensated by sensitivity analysis within a model. Alternatively, modeling limitations can be compensated for by sensitivity analysis across multiple models. That is, several models, such as simulation and scheduling models in the prototype, can be executed based on a single system model to serve as correctives on each other.

A database-centered design thus allows communication between models that address different aspects of the enterprise and that may have different data requirements or be written in different languages. Significantly, this includes simulations which can be dynamically adapted to changing system specifications and results obtained from separately executing scheduling models. Experiments with MMC show the advantages of combining simulations with static and deterministic models in (1) understanding the performance characteristics of the simulated system, and (2) understanding the performance characteristics of the modeling strategies themselves. Users are able to execute models independently or in dynamic combinations, and users can choose to execute models automatically or to intervene in model execution to guide decisions in system design.

The database-centered design preserves a high degree of data independence so that the database schema can evolve and intelligent interfaces can be developed without needing to change models that have already been implemented. Moreover, by emphasizing information relationships, we have begun to establish information structures that will combine simulation with computational models, database management techniques, and, in more recent research, knowledge representation techniques drawn from artificial intelligence.

ACKNOWLEDGMENTS

This research was sponsored in part by Digital Equipment Corporation under research contract KS-995405.

REFERENCES

Kantowitz, B. H. and Sorkin, R. D. (1983). *Human Factors: Understanding People-System Relationships*. John Wiley & Sons, New York.

Ketcham, M. G. (1988). Simulation modeling of just-in-time assembly systems using an information-based methodology. In *Proceedings of the 1988 Winter Simulation Conference* (M. A. Abrams, P. L. Haigh, and J. C. Comfort, eds.) Society for Computer Simulation, San Diego, Ca., 624-628.

Ketcham, M. G., Shannon, R. E., and Hogg, G. L. (1989). Information structures for simulation modeling of manufacturing systems. *Simulation*, 52, 59-67.

Norbis, M. and Smith, J. M. (1986). Two level heuristic for the resource constrained scheduling problem. *International Journal of Production Research*, 24, 1203-1219.

Rochkind, M. J. (1985). *Advanced UNIX programming*. Prentice-Hall, Englewood Cliffs, N.J.

Sprague, R. H. and Carlson, E. D. (1982). *Building Effective Decision Support Systems*. Prentice-Hall, Englewood Cliffs, N.J.

AUTHORS' BIOGRAPHIES

MICHAEL KETCHAM is an Assistant Professor of Industrial Engineering and Operations Research and co-director of the Simulation and Computation Laboratory at the University of Massachusetts, Amherst. His recent research has included the development of techniques for information-based simulation modeling, the integration of simulation with computational models and artificial intelligence in a manufacturing planning environment, and the development of parametric models for simulating electronic components manufacturing and electronics assembly. Professor Ketcham received his Ph.D. in Industrial Engineering in 1986 from Texas A&M University, and is a member of IIE, ORSA, TIMS, and the Society for Computer Simulation.

Michael G. Ketcham
Department of Industrial Engineering and Operations Research
University of Massachusetts, Amherst
Amherst, Massachusetts 01003
(413) 545-2851

RAMESH RAJAGOPALAN received his B.E. in Mechanical Engineering from Guindy University, Madras, India, and his M.S. in Industrial Engineering from the University of Massachusetts, Amherst, where his thesis examined production data models for multiple model communication. He is currently working as an analyst for Norton Corporation in Worcester, Massachusetts.

Ramesh Rajagopalan
Norton Company
1 New Bond St.
Box 15008
Worcester, Mass. 01615
(508) 795-2815