Proceedings of the 2001 Winter Simulation Conference
B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, eds.

COMPUTER-AIDED MANUFACTURING SIMULATION (CAMS) GENERATION FOR INTERACTIVE ANALYSIS--CONCEPTS, TECHNIQUES, AND ISSUES

Boonserm Kulvatunyou
Richard A. Wysk

Department of Industrial and Manufacturing Engineering
Pennsylvania State University
310 Leonhard Bldg. University Park, PA 16802, U.S.A

ABSTRACT

Simulation model is usually developed as a one-time use analytical model by a system analyst (usually from external firm) rather than for a routine and interactive use by a shop floor engineer. This is because it usually takes longer time to generate a result from the simulation, and the simulation model of manufacturing system is usually too sophisticated and time-consuming to use as an interactive tool by the manufacturing/production engineer. A CAMS reduces this complication by encapsulating the 'complicated-logic' and automating the 'tedious data-acquisition' with a more user-friendly interface like a spreadsheet or database input form. This paper describes how CAMS can automatically generate a simulation model; specifically, techniques and issues to structure the model to hide those tasks, so that it is a user-friendly interactive decision support with minimal amount of automation code. The paper concludes with a capacity analysis example from the real industry.

1 INTRODUCTION

Simulation is a powerful manufacturing systems analysis tool that can capture complicated interactions and uncertainties in the system. However,

Performance of a manufacturing system, specifically capacity and throughput, with high product mix is typically difficult to understand. In other words, the performance of a manufacturing system, even in a specific area, is usually affected by system dynamics as bottlenecks move regularly in these systems [Goldratt and Cox 1992]. These dynamics could be the change in product mix, equipment breakdown and maintenance, worker schedules, or even process plans. A normal linear mathematical model can give only a rough performance estimate while ignoring many important factors that may effect the system performance. A simulation model is a sophisticated tool that can take into account many system dynamics. However, a simulation model is usually built for a one-time use by external consultant. This is because production engineers usually spend most of their time prioritizing production on the shop floor using ad hoc solutions (observation).

Furthermore, production engineers though posses analytical skill, they are lacking of either knowledge or interest in the programming task. This habit leaves the sophisticated analytical model, such as simulation model, as a last resource for them. The model usually receives attention only when the manufacturing system having excess-capacity (engineers have free time) that rarely happens in the industry.

The concept of CAMS model allows the simulation specialist to build a sophisticated model that can be easily reused by the plant engineers on a regular basis. The model hides the complicated logic and the tedious data acquisition task from the user. The complicated logic includes the detail that captures the interactions among human and machines as well as production decision logic. The tedious tasks include redefining experimental elements as well as animation elements. The model must be built such that the simulation logic is separated from the system information including manufacturing resource data, process plans, and decision/control parameters. Therefore, the production engineers can interactively change the system information including the production strategies, add or subtract resources, and change in product mix in the Master Production Schedule (MPS) without having to deal with the programming task. The separation of model logic from the system information benefits not only the user side but also the developer side, because it eliminates the automation code required to update the model logic when the system information changes.

Due to the complexity in the manufacturing system and the nature of discrete event simulation, several issues arise when separating system information from the model arise when separating system information from the model logic. The techniques and associated issues from the author experience will be discussed in section 6 of this paper using example scenario described in section 3. Section 2 provides the reviews and comparisons of passed works. Section 4 describes a typical CAMS system. Finally, section 7

summarizes the discussion and emphasizes the value of CAMS concept.

2 LITERATURE REVIEW

Son, Jones, and Wysk (1999) at the National Institute and Standard and Technology (NIST) have developed a methodology for automatically generating simulation models for specific manufacturing scenarios. In this methodology, a neutral language for a specific scenario is developed to semantically describe simulation model. Simulation model is then generated by a model translator, which translates the neutral description into syntax of specific simulation packages. The purpose of the research is to reduce the simulation modeling time as well as to study the issues of transferring a simulation model between simulation packages using a neutral language. The generated model is usually an almost finished model if the model logic is simple or a partially finished (with all necessary experimental elements defined) when the model logic is sophisticated. This is because formally describing a sophisticated behavior of a system is not different from writing a low-level program despite the third generation programming language is available. Therefore, the automatic generation of simulation will not very well facilitate simulation for interactive use. In other words, it is not efficient to generate a new simulation model every time some system information changes. The automatic simulation generation is rather a suitable front-end tool for creating a proprietary simulation model. The neutral language should serve as a back-end tool for archiving and transferring simulation model between software packages.

Simulation model is usually used for studying the system at the design time and is only reused for either as simulation-based control (Smith et. al 1994) or simulation-based scheduling (Harmonosky 1995). There are a number of researches that discuss about using simulation model to understand the system characteristics when the system is in operation. These researches usually focus only on capacity as relating to the number of resources and layouts associated with the material handling. In fact, several other factors effect the capacity, such as process plan and rules governing operational decisions (e.g., what should and can be put in the same batch). These factors are rather very important in a large manufacturing system where uncertainties and high product mix are normal. Off-line interactive simulation is an effective tool to study these kinds of parameters, where system is not fully automated and real-time control does not exist.

Object-oriented simulation has emerged as a prevalent topic in academic research yet slight commercial implementations present (Narayanan et al. 1998). Researchers in this area believe that object-oriented program requires robust abstractions which takes time to develop. Thus, it is not appropriate to a small simulation project. Until user-friendly interface for object-oriented simulation is available, this object paradigm will not be prevalent in this particular industry. The object-oriented simulation is attractive in its mechanism such as inheritance, polymorphism, encapsulation, and more importantly the code reuse of component-based simulation entities. Object-oriented simulation still requires intensive coding; thus, providing apparent benefits (the code reuse) to the developer side more than to the end user side.

3 AN EXAMPLE SYSTEM

3.1 Introduction to the Example Manufacturing System

In this section an example case study from an Alcoa Inc. manufacturing system is described for using in the subsequent discussions. Due to the legal related reason, Nittany Manufacturing (NM) will be used as a facsimile of the real system. In addition, in order to disguise the project of the company, the associated products will remain anonymous.

Nittany Manufacturing (NM) produces products that need to go through a batch furnace annealing process after several upstream processing. The products although have similar shape, they vary by different mechanic of material specifications imposed by different customer needs. Different recipes (or process plans) are generally required to achieve different mechanical properties, which create a high product mix nature in this manufacturing system. However, the products requiring similar recipes can be mixed into the same batch while maintaining the required specification and increasing resource utilization. In other words, changing in the MPS can significantly effect the system performance especially the resource utilization (in this case the utilization is based on resource specifications such as weight or volume capacity rather than the time). There are rules governing with what product specification can be combined into the same batch and what process plan should be followed after the combination. Other constraints such as machine types and capacities are also imposed on the batching process.

Figure 1 illustrates the situation. The furnace capabilities, the combination rules, and the production schedule are given. The production schedule is product A, B, D, and C. Suppose that furnace 1 and 2 are available, and the operator have got ten units of product A and B on the same batch using combination rule #1 for the furnace 1. Subsequently, product D arrives and the furnace operator put all the ten units into furnace 2 since product D can not combine with the first batch according to the combination rule #1. The operator, without knowing that product C is about to arrive, puts the first batch into furnace 1 and begins the process without waiting any longer. When product C arrives, it finds no furnace available because product C can not combine with product D, which already occupies fur-

nace 2. This example shows that more flexible combination rules and larger number of combination rules are better and that the MPS can have significant effect to the system performance. Had product C scheduled before product D, no units would have had to wait.

Furnace capability by product type	Combination rule
Furnace 1: A, B	1. A, B, C run as C
Furnace 2: A, C, D	2. A, D run as D
Processing time by product type	Production schedule
A: 10 hours	A - 10 units
B: 12 hours	B - 10 units
C: 13 hours	D - 10 units
D: 14 hours	C - 10 units

Figure 1: Example Furnace Specifications and Combination Rules

3.2 Problem Statement

Resource capacity becomes an important issue when customer demand surges. Because decision-making tool, time, and collaborative knowledge to analyze this complex manufacturing system are limited, the manufacturing engineer only use rule of thumbs from experience-based observations to correct capacity issues. NM has been facing the late delivery when demand is high, immense inventory expense to stock the product when demand is low, and the profitability dilemma of losing business or investing a large capital to purchase additional resources. In fact, all of these happen while excess capacity still exists even when demand surges. This is evidently shown on the low resource utilization in the historical statistics.

NM's production engineers have been using a simple linear mathematical model where production capacity is a function of production rate multiply by time. The engineers use this model to govern their business decision such as forecasting capacity shortage, generating MPS, and estimating delivery date.

The furnace operators use simple rule of thumbs to set up batch based on the workpiece that they see in the WIP area. In addition to that, furnace operators prioritize tasks based on the worksheet given by the area manager. These rules of thumb are; for example, immediately put hot metal that comes from certain production lines into the furnace, do not wait for more products getting into the batch for so long to start the furnace, and start the process right away if high priority task presents. The priority worksheet is usually only run by the closest promising date. When the 'apparent' capacity seems very tight, the manufacturing system engineer then steps down to the floor to prioritizing how things are moved, squeezing the furnace space, postponing 'unnecessary' activities (such as schedule furnace maintenance, changing thermocouple, etc).

These activities seem to go on throughout the year because there is no single analytical tool where engineers can share their different aspects of common task (optimizing the throughput). The *production engineer* plans on the master production schedule without knowing any equipment or personal breakdowns (system status). The *manufacturing system engineer* simply uses experience-based judgement, e.g., "I have it done this way last time and it was better so that should do it again this time." When the scenario looks the same (capacity shortage), but the input maybe different. The *process engineer* tries to improve the process plans and annealing recipes without knowing the actual effect to the manufacturing system. *Sales and other engineers* are not updated and are still using the same old numbers for crunching the production plan.

Regarding to the problems described above, NM's industrial engineering team has made a recommendation in a regular plant meeting that a sophisticated decision support tool is in need to solve these ongoing problems. This tool must be capable of capturing the interaction of the factors that effect actual capacity in the batch furnace area. The tool should also help in making sound analytical judgement governing planning and execution of production in the batch furnace area. Furthermore, it should enable the information and task integration of effecting engineering departments. That is acquiring data from the common pool of data such as an MRP or ERP system.

NM's management agrees on the proposal and has amended an additional use of the tool in the ongoing effort to reengineer NM's production system to minimize inventory cost and increase agility (the ability to cost-effectively respond with smaller lot sizes of customer orders). One methodology considered in NM's production system is the flowpath redesign, which is gearing toward minimizing material handling cost and time. As an interactive tool, the suite should allow engineer to analyze the effects of some of these NM production system implementations.

3.3 Simulation Model Specification

In the next plant meeting, the engineers have agreed as the first step that the tool should capture the actual operations performed by the operators from setting up a batch until unloading it from the furnace. The tool will allow the user to parametrically modify those operations and heuristically analyze the effect on the system performances. Examples of parameterized operations in the described system are the maximum period the operator should wait for more products that can be put into the same batch and the minimum furnace capacity that the batch should fill up before starting the annealing process. In addition, the tool should allow the user to interactively changes system definition so that extensive 'what-if' analysis can be done and the simulation result can be easily validated. These system definitions include the number of resources (both human and machine), master production schedule (input mix and product arrival process), and process plan. It happens in a

particular manufacturing system that the raw material is always more than enough; therefore, the material requirement planning is excluded from the consideration.

At the end of the meeting, the engineers conclude that simulation model is an appropriate tool since it is known as an effective tool to capture the complex interaction in the manufacturing system. In order for the simulation to facilitate the desired "what-if" analysis, the system definitions and interested parameters should be separated from the model logic and stored in a user-friendly database system. That is the computer-aided manufacturing simulation generation is desired.

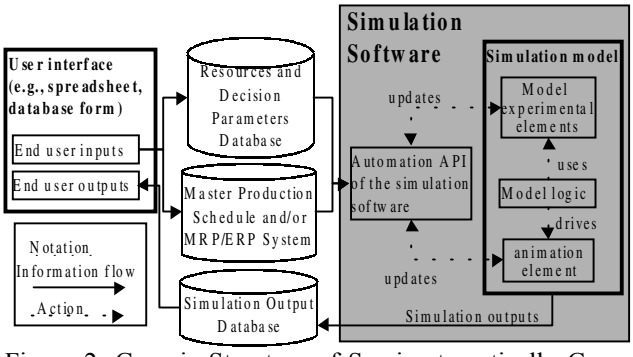

Figure 2: Generic Structure of Semi-automatically Generated Simulation

4 THE STRUCTURE OF CAMS

The purpose of CAMS is to allow the user to interactively conduct analysis by modifying the system definitions, master production schedule (MPS) and interested decision parameters. In addition to facilitating the analysis, the CAMS hides the programmatic complexity from the user and presents only spreadsheet like user-interfaces for the user to define input data and analyze the result.

Figure 2 illustrates the generic structure of CAMS. The most important difference of CAMS from the automatically generated simulation (Son, Jones, and Wysk 1999) is that CAMS does not generate the model logic. As mentioned earlier in the literature review section that the automatically generated simulation model is usually incomplete because formally and completely describing model logic, using high-level language is not simpler than syntactically describing with the low-level language. Hence, CAMS skips that attempt as illustrating in Figure 2 that there is no connection between the automation code and the model logic as well as the one way action from the model logic to the experimental and animation elements. This structure results in a significant reduction in the amount of automation code and a modular architecture, which allows the software to be quickly developed and updated. CAMS model however allows the model logic to be parametrically modified through the experimental element definitions.

With the database integration, the simulation output can be written back to the database. This output can be then customarily formatted and presented to the user to fit their specific analysis perspectives. Although output presentation feature is available within some simulation packages (such as ARENA), performing presentation task in the database software is much more flexible and vivid.

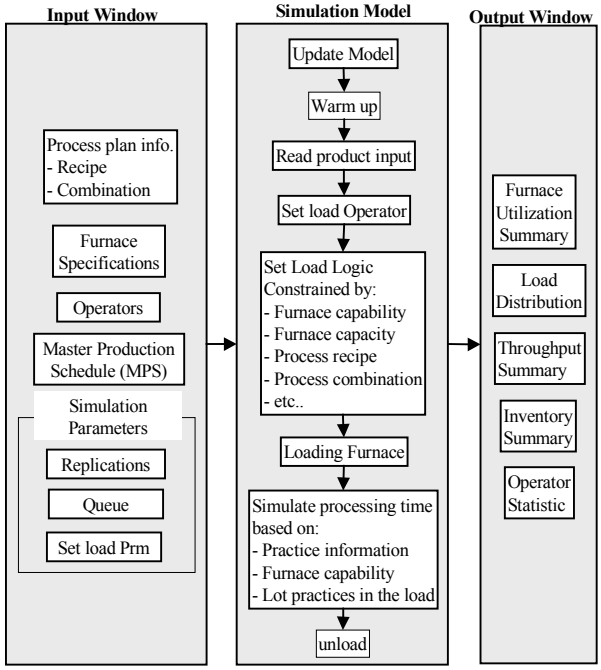

Figure 3: Nittany Manufacturing's Computer-aided Manufacturing Simulation System Description

5 CAMS MODEL EXAMPLE

Figure 3 illustrates the components of CAMS model that was created for the NM's batch annealing area. The user interface and data storage was implemented in the Microsoft Access. The simulation model was developed in the ARENA simulation modeling software. The Visual Basic for Application (VBA) was used to code the automation that links Microsoft Access and the ARENA. Although ARENA provides several apparent advantages toward an interactive simulation analysis tool such as the VBA linkage to other office suites and the availability of aggregate type elements, other software suits were not evaluated. Therefore, no inclination should be implied. The advantages however may not be available in every simulation package or maybe available in different forms. The rest of this section will provide an insightful detail of the interesting points in the Input window and the Simulation model window that are illustrated in Figure 3. This detail will reference the discussion of techniques and issues in the next section.

At the top of the input window (Figure 3), the process recipes and batch combination rules maybe adjusted by process engineer who works on the recipe simplification so that more combinations are possible. More combination

possibilities mean more types of products can be combined into a single batch to fill up the furnace. However, the products combined into the same batch may go through unnecessary processing steps. This can be seen from the example of combination rule #1 in Figure 1. Products A, B, and C can be combined using that rule and processed as C, yet product A and B, which normally would require only 10 and 12 hours of processing time, would be processed for 13 hours. Process engineers need this simulation tool to justify the possible adverse effect of the changes they are trying to make to the system performance.

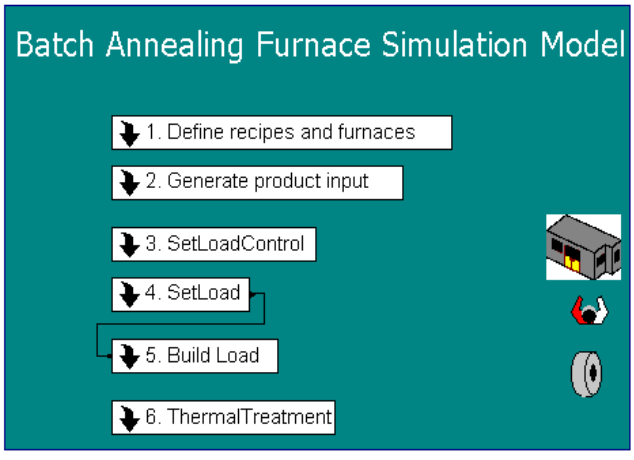

Figure 4: High-level Simulation Models

The furnace specifications in the input window stores information such as furnace capability (e.g., the type of product the furnace can process), furnace capacity (e.g., how big of a batch the furnace can handle), and how long it takes to process a particular batch (e.g., heating rate). The user can also define operator schedules and simulation parameters. It should be noted that the MRP/ERP integration is excluded from the model since it is beyond the scope of this paper and the company does not have it available at this time. The simulation parameters includes the experimental definitions (e.g, terminating condition, queue rules) and control logic parameters. In this particular problem, the interested control logic parameters are the set load parameters, which may either control the minimum batch size or maximum waiting time.

The simulation model for this example is subdivided into six submodels (Figure 4) as follows:

1) Define recipes and furnaces --read the system definition and the MPS then re-generate the model accordingly
2) Generate product input --create product entities according to the MPS
3) Setload control --initialize the set up batch activity
4) Setload --find products to fill the batches
5) Build load --put together selected products into a batch
6) Thermal treatment --process and unload the batch and store statistics

Only the 2) Generate product input, 3) Setload Control, and 4) Setload submodels are delineated here as they provide good examples for the techniques and issues described in the next section.

5.1 Generate Product Input Submodel

The resource and product specifications are read from the database into the simulation memory in the first submodel (this submodel is not illustrated here). The specifications specify resource capability, product sizes, and recipe requirements. The second submodel (Figure 5) generates entities representing the product according to the MPS and the arrival process. It is assumed in the initial release of the tool that the user specifies the MPS by indicating the daily mix of product inputs for a week and that the products scheduled to produce in a day arrive at the beginning of that day (i.e., every 24 hours).

The first block of this submodel creates a dummy entity every 24 hours. The entity repeatedly goes through two nested WHILE loops to trigger the DUPLICATE block. Each pass through the DUPLICATE block, the dummy entity creates a number of product entities equal to the number products coming from an upstream location requiring a particular recipe (these numbers are once again user inputs). The number of times entity repeating the duplication equals to the amount of product inputs for that day. The generated product entities are stored in the logical queue, named WaitingForSetloadQ, which is a logical place where products are waiting to be selected and loaded into the batches. In this submodel the dummy entity is a *control entity* which is only created to trigger the product entity creations

5.2 Setload Control Submodel

The Setload control submodel initiates a set load activity when the furnace is empty. This submodel has the number of control entities equals to the number of furnaces in the system. Each entity looks over the furnace availability. When the furnace is available, the control entity triggers the VBA block 3 to search for the product entity in the WaitForSetLoadQ (Figure 6) that is applicable to the furnace capability. Heuristics are used in the search logic to choose the type of product when there are more than one type found that requires the same recipe and is applicable to the same furnace. The heuristics used maybe 'selecting the product type where the maximum number of product entities is found'. The product entity found is sent to the next submodel, Setload submodel, to find more product entity for the batch.

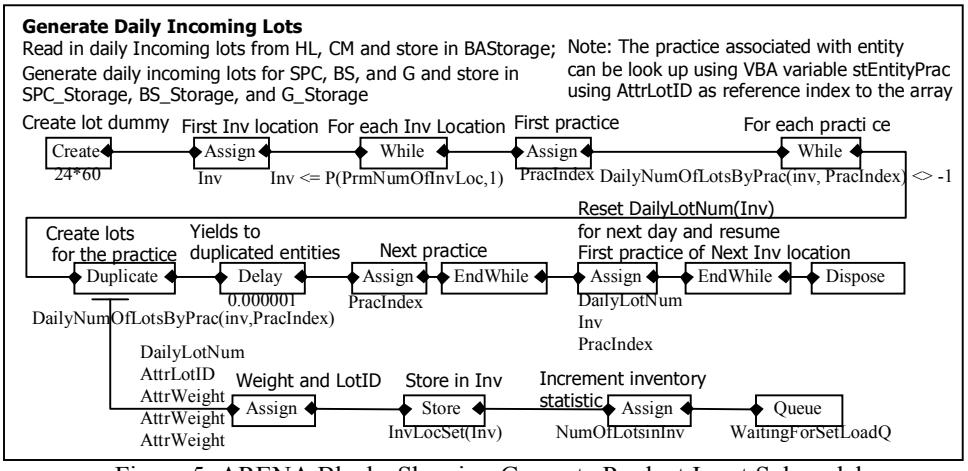

Figure 5: ARENA Blocks Showing Generate Product Input Submodel

5.3 Setload Submodel

Setload submodel (Figure 7) uses the active product entity to find the next product entity to put into the same batch. If the active product entity can find the *next product entity*, the active product entity is put into the bucket of the associated furnace (logical queue of each furnace, imagine the operator noting down on the paper each workpiece he found that can be put in the same batch). The *next product entity* then becomes an active entity and finds another product entity repeatedly until the furnace is filled or other control strategies are matched (e.g., max waiting time). There are also some heuristics associated with the search for the next product entity such as fill the batch with products which require the same recipe before products which are in the same combination rule. This is in effect because when products are combined, the furnace usually must follow the recipe of the one having longest processing time.

6 MODELING TECHNIQUES AND ISSUES

Parameterizing and modularizing the model logic require some special techniques. The techniques and issues discussing here rely on the ARENA's entity-driven and event-based simulation engine (Note that some simulation packages may be resource-driven and process-based, where resources are polling for entity).

6.1 Using Aggregate Types of Elements

Aggregate type elements are very important features to modularize the simulation logic from the model definition (simulation elements). The aggregate element can be a placeholder of any type of element and thus the model logic can refer to the individual element by just using the indices. There are two aggregate types in ARENA, which are SET' and ARRAY. The set can contain most of the experimental elements such as resources, storage areas, queue, etc. On the other hand the array can contain only primitive type variables.

An example use of the SET element is when resources, queues, and storage areas are organized into sets. Adding (or subtracting) resources will not require additional control logic modules to regulate the behavior of each of them. By carefully design the control logic modules, the same modules can handle the change using resizable member index. Furthermore, ARENA's entity can seize a set of resources according to specified conditions (e.g., cyclical, min utilization); thus, the additional resources just need to be added to the set and upper bound index is resized.

Array is a very common feature to all programming languages. ARENA array feature has limited capability in that the array size can not be changed during the simulation run. The array dimension also can not be parameterized (must be declared with a numerical literal).

6.2 Using Parameter and Variable Elements

Since most of the information used in the interactive CAMS must be parameterized, most of the logical statements must be based on the parameters and variables. Frequently, there is some confusion between the usage of PARAMETER and VARIABLE elements as they are programmatically identical (having single reference throughout the simulation run), yet having different usage. The parameter typically maintains its value throughout the simulation run, while the variable is usually updated throughout the run. The values such as decision criteria and number of resources are good examples of the parameter element usage, since they are *system definition*.

ARENA's VARIABLE and PARAMETER elements have a limitation in that they can not hold string literal. Consequently, any string information can not be directly handled. The simulation programmer will need to either index the string with numerical value or manipulate the string using the VBA module, which may cause the model to be cluttered and hard to follow and modify.

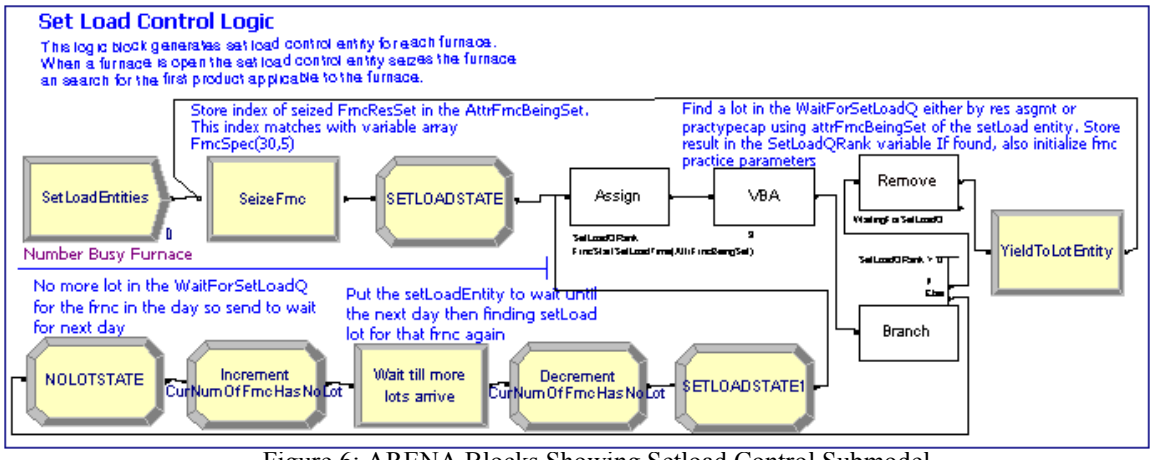

Figure 6: ARENA Blocks Showing Setload Control Submodel

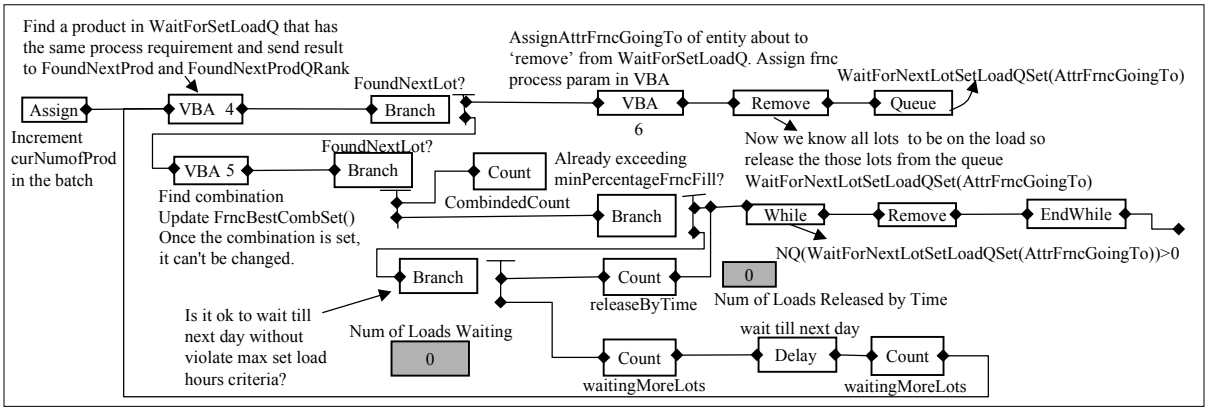

Figure 7: ARENA Blocks Showing Setload Submodel

6.3 Conditional Wait and Logical Queue

Generally the conditional wait and logical queue are handy in the simulation model that needs to capture control logic of a manufacturing system having a number of interactions between resources. It should be noted that a logical queue may not represent a real queue in the manufacturing system and is mainly used for decision making and controlling purposes. In ARENA, a logical queue is called DETACHED queue. The entities in this queue proceed only when another entity issues a command to do so. An example use of conditional wait is when two or more resources must be available at the same time, until then the waiting entity can proceed. In ARENA, this can be implemented several ways. One way is having the entity wait until the condition (number busy at the required resources are zeros) becomes true. Other ways are having another entity, which must know when all the required resources are available (or all pre-conditions are met) sending a signal to the waiting entity or sending a REMOVE command to the waiting entity. In our example model, detached queues are used in a few of places. One is where the entities are waiting for setting up a batch at the end of the Generate product input submodel (Figure 5). The other is where the entities are waiting for the batch to be filled in the Setload submodel (Figure 7).

Care must be taken in using these conditional wait and logical queue. In the conditional wait, the programmer should be certain that the simulation software behave as he or she expects, especially how the condition is evaluated. The condition in the ARENA's conditional queue (SCAN or HOLD block) is only evaluated before every next discrete time advance. Therefore, it is possible that the condition becomes false again by another entity before the waiting entity see the condition when it is evaluated to be true.

The conditional waiting that relies on signaling or a REMOVE command seems to guard against the above problem. However, another care must also be taken to which entity is proceeding first after the signaling or issuing of the REMOVE command. In ARENA, the entity sending the signal or issuing the command continues its execution and the entity receiving the signal is only placed at the top of the event calendar. In our example model, we usually want the removed entity to proceed before the signaling entity, which will set/reset the control or index variables. This is where we need to delay the signaling entity with an infinitesimal amount of time so that the simulation looks up the event calendar and the removed entity is

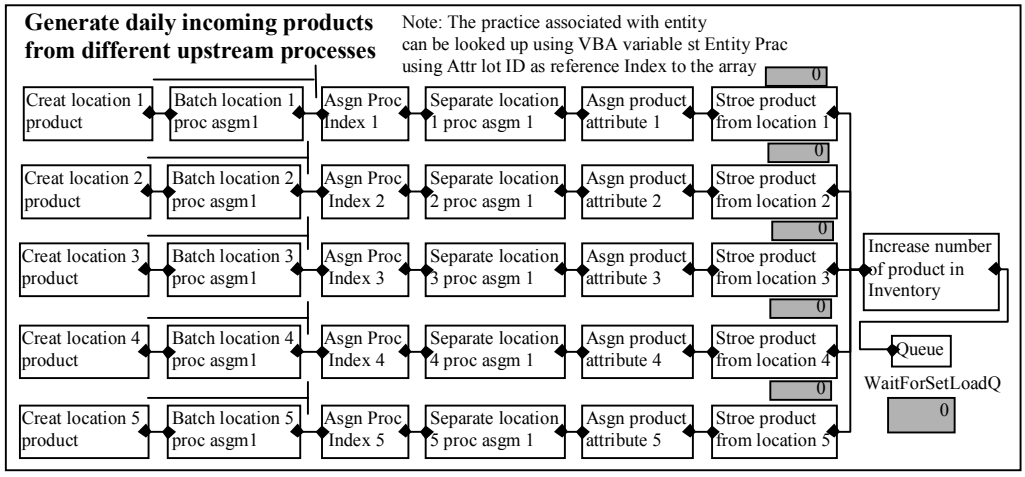

Figure 8: Generate Product Input Submodel that does not Use Control Entity

yielded. Moreover, care must be taken that the infinitesimal delay does not effect the simulation result. This is sometimes very difficult to check at the design stage, but easy to detect when verifying the model. Generally, if the small delay is undisputed, slightly changing the delay period (for example from 1E-6 to 1E-5) should not result in a noticeable difference in the system performances. The dummy entity in the Generate product input submodel uses this technique to yield to duplicated product entities and the setload control entity uses this technique to yield to the product entity removed from the WaitForSetLoadQ.

6.4 Using Control Entity

In order to modularize the control logic with respect to the model experimental elements such as the number of resources, control entity is particularly useful (control entity is an entity, which may or may not represent a physical object in the modeling system. An entity is a control entity when it is used to control other entities or elements in the model). Take the Generate product input and the Setload control submodels as examples. Without using the control entity the Generate product input submodel might look like the ARENA blocks shown in Figure 8. In order to make the submodel flexible to additional upstream locations, each of the five branches in the submodel handles the generation of product entities from each upstream location. Without using the control entity technique, a piece of automation code will be required to automatically add a similar branch of the ARENA blocks to the submodel for each additional upstream location.

Similarly, haven't the setload logic configured into two submodels (the Setload control and the Setload submodels) using control entities (each responsible for each resource), and the aggregate elements, the setload logic might be cluttering like the submodel shown in Figure 9. It should be noted that the product entity in the Setload submodel (Figure 7) itself functions like a control entity searching for the next product entity. In Figure 9, each loop handles each resource, which may have different specification. An additional loop will be needed for each additional resource. The loop may be much larger as each of them requires a chunk of logic blocks to handle the algorithm for searching the product entity. The submodel is quickly cluttered only when a few resources are in the system.

Care must also be taken when using control entity as the simulation speed and the result may be effected. For example, take a look at the block labeled 'Wait until more products arrive', which is a DELAY block, in the Setload control submodel (Figure 6). The control entity is delayed at that block when the first product entity for the new batch can not be found in the current day, given an available furnace. In the example model, it was assumed that the products only arrive at the beginning of each day. Therefore, the control entity can be delayed for the elapse time between current time and the beginning of the next day to start the search for applicable product entity again. This deterministic time delay maintains an acceptable simulation speed. However, if the product entity arrival time is a random number, the control entity needs to continuously search for an applicable product entity. On the other hand, if there is no holding block in the control loop (note that the entity encountering a holding block causes the simulation to look up the event calendar for the next scheduled event. Entities are held when they encounter commands like delay, seize, and wait), the control entity will not stop its execution and other entities in the model can not proceed. The optimal delay time in this case is the elapse time between current time and the schedule arrival time of the next product entity. If this time periods are very short, the simulation need to stop to advance the clock more frequently which will result in significant reduction in the simulation speed. Note that similar problem also applies to the Setload submodel when the active product entity does not find the next applicable product entity in the WaitForSetLoadQ queue, it has to yield to other entities.

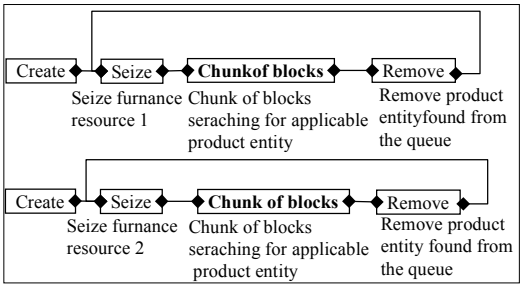

Figure 9: Cluttered Setload Submodel

7 CONCLUSION

The concept of the Computer-Aided Manufacturing Simulation (CAMS) for interactive analysis has been introduced. The concept relies on the parameterized and modularized model logic that allow the end user to interactively modify the model and analyze the effect without having to deal with the complex logic. The CAMS tool should yield the following benefits to the organization. The shop engineer can spend more time and attention to figuring out the manufacturing and production problems rather than programming problems. The CAMS tool should help saving time and money from hiring a consultant every time some system definition changes or an analytical question arises. In addition, the modular architecture allows engineers from several departments to integrate their information into a common portal and consequently always obtaining an up to date analysis result.

An example manufacturing system and associated CAMS model has been given. The techniques and issues that help making the model logic modular and parameterized are described using the example model of a batch processing problem. Though the breadth of modeling technique is unlimited, similar issues will be arising. The challenge to the simulation programmer is to modularize the simulation model while maintaining the simulation speed and validity. The Nittany Manufacturing's batch annealing analysis tool, as shown in this paper, is useful for giving the engineer a quick quantitative answer to whether what they do is better or worsen the system performance. In the future, an optimization tool can be plugged-in (ARENA 4.0 now offers the OptQuest optimization software integration), the tool will be able to provide short-term heuristically optimal strategy. While the explanation in this paper is based on a single example, the concepts, techniques, and issues should apply to any type of application.

REFERENCES

Goldratt, E.M., Cox, J. (January 1992). The Goal: A Process of Ongoing Improvement North River Press, Incorporated.

Harmonoski, C.M. Simulation-based real-time scheduling review of recent developments. Proceedings of the 1995 Winter Simulation Conference, 220-225.

Narayanan, S., Bodner, D.A, Sreekanth, U., Govindaraj, T., McGinnis, L.F (1998). IIE Transactions 30, 795-810.

Narayanan, S., Malu, Pallavi G., Ashili, Krishna P.B., Di Pasquale, John, Carrico, Todd M (Dec 1998). Web-based interactive simulation architecture for airbase logistics systems analysis. International Journal of Industrial Engineering : Theory Applications and Practice, 5 (4), 324-335.

Smith, J. S., Wysk, R. A., Sturrok, D. T., Ramaswamy, S. E., Smith, G. D., and S. B. Joshi (1994). Discrete Event Simulation for Shop Floor Control. Proceedings of the 1994 Winter Simulation Conference, 962-969.

Son, Y. J., Jones, A.T., Wysk, R.A. (2000), Automatic generation of simulation models from neutral libraries: An Example. Proceedings of the 2000 Winter Simulation Conference.

AUTHOR BIOGRAPHIES

BOONSERM KULVATUNYOU is currently a Ph.D. candidate in the Harold and Inge Marcus Dept. of Industrial and Manufacturing Engineering at Pennsylvania state University. He received his MS from Columbia University and his BS from Chulalongkorn University, Bangkok, Thailand. He is a member of the Society of Manufacturing Engineers' Student Chapter at Pennsylvania State University. His research interests include computer-integrated manufacturing system simulation, and information modeling.

RICHARD A. WYSK currently holds the Leonhard Chair in Engineering at the Pennsylvania State University. Prior to his current position, he was director of the Institute for Manufacturing Systems and holder of the Royce Wisenbaker Chair in Innovation at Texas A&M. he has also served on the faculty of Virginia Tech and worked in industry as a research analyst for Caterpillar Tractor Co. and as production control manager for General Electric. He received his Ph.D. in industrial engineering from Purdue University in 1977 and his BS and MS in industrial engineering and operations research from the University of Massachusetts (Amherst) in 1972 and 1973, respectively. He is a decorated Vietnam veteran and author of several textbooks. Honors recognizing his research include the Institute of Industrial Engineers' David F. Baker Distinguished Research Award and the Society of Manufacturing Engineers' Outstanding Young Manufacturing Engineering Award. His research interests include computer-integrated manufacturing, computer-automated manufacturing, computer-aided process planning, and concurrent engineering.