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

A KANBAN MODULE FOR SIMULATING PULL PRODUCTION IN ARENA

Mark A. Treadwell
Jeffrey W. Herrmann

Institute for Systems Research
University of Maryland
College Park, MD 20742, U.S.A.

ABSTRACT

In the short timeline of rapid improvement events (kaizen events), it is difficult to use the full power of simulation because of the time required to construct models, particularly if the system uses pull production control methods such as kanbans. This paper describes multiple ways to model pull production control and compares them on measures related to model construction and execution. A kanban workstation module significantly reduces the time required to develop a pull production model, which makes simulation more useful as a decision-making tool in rapid improvement events.

1 INTRODUCTION

One popular tactic in the lean manufacturing "toolbox" is the rapid improvement, or kaizen, event. A rapid improvement event comprises a highly compact sequence of activities: examining the current conditions, identifying potential areas of improvement, and implementing the proposed changes. Usually these events take place over the course of a single week, with several weeks of preparation beforehand and a follow-up analysis period afterwards.

Since facilities are placed back into production immediately following the conclusion of the event, it is generally not feasible to extensively test a wide range of system configurations. In this situation, simulation can be extremely helpful for investigating alternative designs; however, given the compressed timeline, it can be difficult to produce a helpful model. This problem is exacerbated when one considers that kanbans and other pull production control methods are frequently implemented to limit work-in-process inventory. The extra complexity of pull production control increases model construction time even further.

This paper focuses specifically on kanbans, which are a popular method for implementing pull production control in manufacturing cells. A kanban is a card or some other mechanism that authorizes the workstation to produce a part. When the workstation completes the part, the kanban stays with the part in the downstream workstation's input queue (or buffer). When the downstream workstation begins working on the part, the kanban is released. In a simple production line, raw material waiting to be processed at the first workstation has no kanbans. At the last station, a part's kanban is released as soon as the part leaves the station (so that the last station is always authorized to work).

Manufacturing cell design processes are used by many firms, as described in Suzaki (1987), Suri (1988), Rother & Harris (2001), Conner (2001), Hales & Andersen (2002), and Hyer & Wemmerlöv (2002). For more about pull production control and kanbans, see Hopp and Spearman (2001), Askin and Goldberg (2002), and Black and Hunter (2003).

Modern discrete-event simulation software has many modules to help analysts quickly construct simulation models of manufacturing systems. These include stations, conveyors, and guided transporters, to name a few. Moreover, simulation is a very useful tool for designing manufacturing systems. See, for instance, the survey by Smith (2003).

Pull production control, however, has not been adequately addressed. This paper discusses various ways to implement pull production control in discrete event simulation (specifically, Arena, by Rockwell Software) and describes work done to compare the construction effort and execution time of models built using these alternative approaches. A kanban workstation module was found to be the best approach. In this approach, kanbans are modeled as resources. Thus, all of the overhead for tracking kanbans is managed internally, while providing flexible options for entity creation or input from an external source. The analyst needs to include only a single, easily customized module to represent each workstation. Constructing a complete model then requires much less effort than if it were to be constructed from standard Arena components. At this point, the kanban workstation module can model stations with parallel, identical servers but cannot be used for fork or join processes.

Gahagan and Herrmann (2001) identified the need for adaptable simulation models for evaluating different production control policies. They described a general framework that covers a wide variety of production control policies, not just kanbans. Williams, Ülgen, and DeWitt (2002) created a kanban simulator that uses an Excel spreadsheet as the interface to manage the parameters of a Witness simulation model. Their system provides sophisticated modeling of storage and transportation systems, allowing for automatic optimization of the number of kanban cards and tracking of inventory levels. This approach, while offering excellent support for in-depth analysis, was not designed for the quick construction of simulation models during a rapid improvement event.

The rest of this paper is organized as follows: Section 2 describes the alternative modeling approaches. Section 3 discusses the design of the experiments that were performed. Section 4 presents the results, and Section 5 concludes the paper.

2 MODELING ALTERNATIVES

Using a modern simulation software package such as Arena gives an analyst a great deal of flexibility. Consequently, there are many different possible ways to model a workstation using kanbans. In this work, we focused on two concepts: modeling kanbans as *entities*, and modeling kanbans as *resources*. These concepts were implemented in various ways, as described below.

The first model examined was one constructed previously to model a kanban pull production line. This model used entities to represent the kanban cards and operators, and had them flowing through the system alongside the parts. Operators, cards, and incoming parts would wait in queue until all three were available at one station, and then incoming parts were transformed to kanban cards and returned to the previous station while a new card took on the logical identity of the part. This strategy is probably not the best way to model a kanban system; while it does accurately represent the functionality of the system, the logic within the model is very difficult to follow, as shown in Figure 1. The conceptual representation is also very complicated, as no single entity exists to symbolize a part moving through the system; each part is represented by a different kanban card for each station it passes through. The model can be made more efficient by removing the entity type and entity picture assignments which follow each process, but this does not change the logic governing its function. The simplified model type will be referred to as **Entity** in this paper, and the original version as **Entity2**.

Since a limited number of kanban are available at each workstation, and they are only accessed by a single entity at any one time, the modeling of the system can be simpli fied by using resources to represent kanban cards. This does not affect the logic behind the model, but lets Arena handle all the work involved in managing the cards. This model type will be referred to as **Resource**; its governing logic is shown in Figure 2.

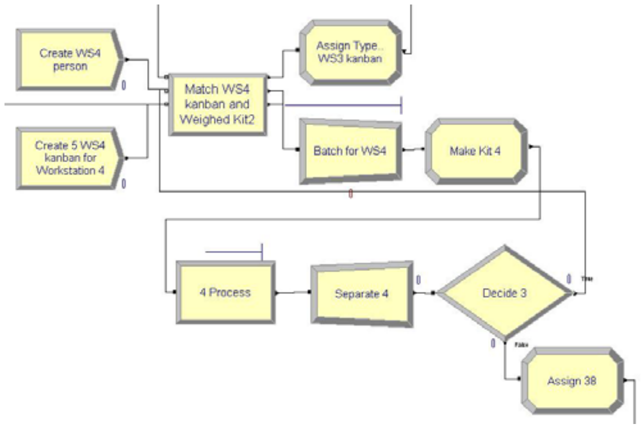


Figure 1: Logic for a Normal Workstation in the Entity2 Model Type.

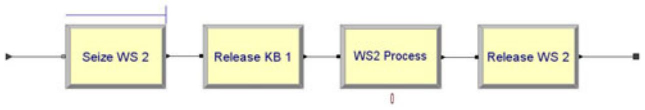


Figure 2: Logic for a Normal Workstation in the Resource Model Type

The high-level constructs that Arena users work with are made up of low-level blocks and elements. These correspond directly to the Siman code produced to run the simulations; in order to run a model, Arena must translate modules and data into these component units. Building a model with blocks and elements reduces the amount of time required to compile and run a model; this is referred to as the **Resource Block** model type.

Finally, the logic of the Resource Block model type was used to create a kanban workstation module that a user can easily manipulate while modeling complex systems. The kanban workstation module includes a dialog that encompasses all information necessary to define a workstation, including a name, its position, an expression for processing time, and the number of kanban available at the station. The main departure from the Resource Block model type is that the kanban workstation module contains logic to handle several different situations, for instance stations at the beginning, middle, and end of a kanban production line. These stations employ most of the same logic but each have slightly different requirements; the kanban workstation module incorporates all three, with several alternate logical paths to be selected based on the user's choice in the module customization dialog. The logic is pictured in Figure 3 below, with boxes drawn to denote the various paths; the customization dialog is shown in Figure

4. The rest of the paper will refer to this as the **Module** model type. The Arena template file is available online at the following URL:
<http://www.isr.umd.edu/Labs/CIM/projects/lean/kanban/>.

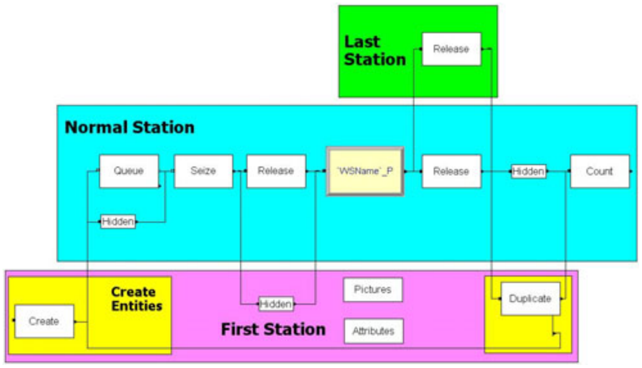
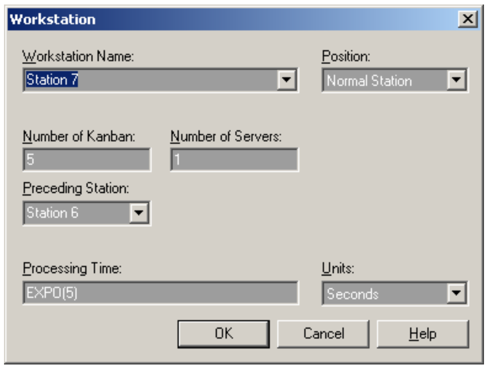


Figure 3: Logic Defining the Kanban Workstation Module.



Workstation Name: Station 7
Position: Normal Station
Number of Kanban: 5
Number of Servers: 1
Preceding Station: Station 6
Processing Time: EXPO(5)
Units: Seconds

Figure 4: User Interface of the Kanban Workstation Module.

3 EXPERIMENTAL DESIGN

The purpose of the experiments was to determine how the different model types affected the effort needed to construct a simulation model and the computational effort needed to run the model.

To measure the effort required for the user to create and customize a workstation, we counted the number of user actions to construct a typical station. Three workstation positions were examined: a *beginning workstation* that generates parts, assuming an infinite supply of raw material, and sends them to another kanban workstation; a *normal workstation* that has kanban-enabled workstations before and after; and an *end workstation* that produces completed parts (there is no workstation afterwards). For each of the model types described above, user actions were counted for constructing a first station, constructing a normal station using new modules, constructing a normal station by modifying a copy of an existing normal station, constructing an end station with new components, and modifying an end station from an existing normal station. Table 1 presents a sample list of the user actions required to create a new normal station in the Resource model type. Individual modules in the model are listed on the left, with the required customization steps on the right.

Table 1: User Actions Required for Station Creation

Normal Station	**(From scratch)**
Seize	Create Connect to previous Name Add resource Resource name Add resource Resource name
Release	Create Connect to previous Name Add resource Resource name
Process	Create Connect to previous Name Action Delay type Delay units Delay expression
Release	Create Connect to previous Name Add resource Resource name

To measure computational effort, we generated the Siman code for each model and counted the number of statements produced for each station position. We also ran the models and measured the time needed to execute multiple replications.

To verify the effects of the different model types in a realistic setting, several different models were tested. An existing simulation model of a nine-station cartridge assembly line represented a typical scenario for a rapid improvement event; a three-station assembly line served as a short model; and a rebalanced version of the short model demonstrated the effects of including multiple servers at a station. Each model was adapted to use the five model types described in Section 2; the average time of ten runs from calling Arena to its return was calculated, where each run consisted of ten replications of 100 hours each.

4 EXPERIMENTAL RESULTS

Figure 5 portrays the effort required to construct stations in various positions for each type of model. The Module model type consistently requires significantly less effort than the other model types, ranging from 50% fewer user inputs to 90% fewer.

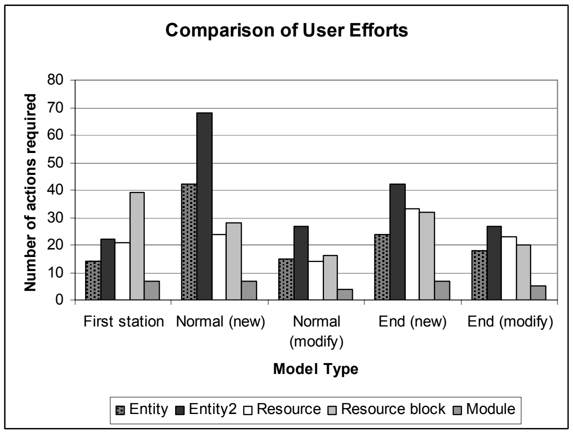


Figure 5: Comparison of Effort Required for Various Model Types and Workstation Positions.

Table 2 gives the number of statements produced by each model type. While the Resource Block model type generates the smallest number of statements, the Module model type is not far behind.

Table 2: Number of Statements Generated by Various Model Types for Several Workstation Positions.

Model Type	**Workstation Position**		
	Beginning	**Normal**	**End**
Entity	15	28	21
Entity2	16	32	23
Resource	15	13	19
Resource Block	6	5	6
Module	9	8	10

Figure 6 plots the number of statements for all three workstation positions in each model type against the number of user inputs. In both cases, smaller numbers are better; it is clear that the Module model type does a better job on both performance measures.

The mean runtimes observed for each model type, applied to each of the three models, are listed in Table 3 below. Since runtimes are directly related to the number of statements that must be processed, it is not surprising that the Resource Block model type has the lowest times. However, the Module model type is close behind.

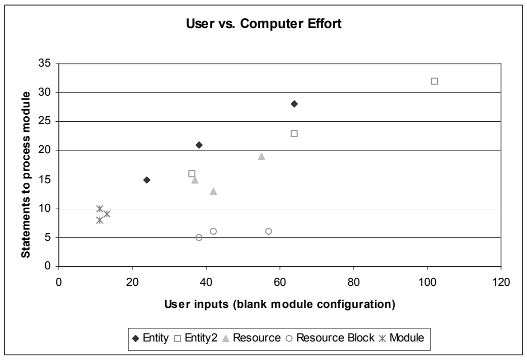


Figure 6: Comparison of User Effort to Number of Statements Produced.

Table 3: Mean Runtimes for Various Model Types, Applied to Three Models.

Model Type	**Model**		
	Simple	**Cartridge**	**Multi**
Entity	4	11.3	5.3
Entity2	4.5	12.2	5.9
Resource	3.9	7.5	5.3
Resource Block	2.1	3.8	2.7
Module	2.5	5.1	4.3

5 CONCLUSIONS

The results detailed in the previous section have demonstrated that the kanban workstation module achieves our goal of reducing user effort. Based on the number of inputs required, a user building a simulation model with this tool should be able to complete initial construction in less than half the time he or she would have required when using conventional means. This savings is accomplished mainly by automating the process of filling in repeating parameters such as the module name, and allowing Arena to manage naming and tracking of all the required resources.

Comparisons to other model types demonstrate that use of the module does not increase the computing time required to run a model; in fact, models built with the kanban workstation module ran faster than all of the other models except those built from the most low-level model type (which is, correspondingly, the least user-friendly as far as the interface goes). Runtime savings are on the order of 33 to 50% for a fairly large model with nine stations; the decrease is only a few seconds in this case, but for even bigger models the difference becomes more significant.

The targeted use of this simulation construction tool is the kaizen rapid improvement event; due to the short timeline and frequently complex models involved, any time that can be saved in the construction of a simulation model can make a significant difference in the overall success of the project. The ability to create and adapt simulation models quickly and easily makes it possible for more concepts to be evaluated in a short time. Simulation can then play a larger role and provide better support for decision makers, which will improve the outcome of the project.

REFERENCES

Askin, Ronald G., and Jeffrey B. Goldberg. 2002. *Design and analysis of lean production systems*. New York: Wiley.

Black, J.T., and Steve Hunter. 2003. *Lean manufacturing systems and cell design*. Dearborn, Michigan: Society of Manufacturing Engineers.

Conner, Gary. 2001. *Lean manufacturing for the small shop*. Michigan: Society of Manufacturing Engineers.

Gahagan, Sean M., and Jeffrey W. Herrmann. 2001. Improving simulation model adaptability with a production control framework. In *Proceedings of the 2001 Winter Simulation Conference*, ed. B.A. Peters, J.S. Smith, D.J. Medeiros, and M.W. Rohrer, 937-945. Arlington, Virginia: Institute of Electrical and Electronics Engineers.

Hales, H. Lee, and Bruce Andersen. 2002. *Planning manufacturing cells*. Dearborn, Michigan: Society of Manufacturing Engineers.

Hopp, Wallace J., and Mark L. Spearman. 2001 *Factory physics*. 2nd ed. Boston: McGraw-Hill.

Hyer, Nancy, and Urban Wemmerlöv. 2002. *Reorganizing the factory: Competing through cellular manufacturing*. Portland, Oregon: Productivity Press.

Rother, Mike, and Rick Harris. 2001. *Creating continuous flow: An action guide for managers, engineers and production associates*. Massachusetts: The Lean Enterprise Institute, Inc.

Smith, J.S. 2003. Survey on the use of simulation for manufacturing system design and operation. *Journal of Manufacturing Systems* 22 (2): 157.

Suri, Rajan. 1998. *Quick response manufacturing: A companywide approach to reducing lead times*. Oregon: Productivity Press, Inc.

Suzaki, Kiyoshi. 1987. *The new manufacturing challenge: Techniques for continuous improvement*. New York: The Free Press.

Williams, Edward J., Onur M. Ülgen, and Chris DeWitt. 2002. *An approach and interface for building generic manufacturing kanban-systems models*. In Proceedings of the 2002 Winter Simulation Conference, Volume 2, eds. Enver Yücesan, Chun-Hung Chen, Jane L. Snowdon, and John M. Charnes, 1138-1141.

ACKNOWLEDGMENTS

The authors wish to thank the personnel at Naval Surface Warfare Center, Indian Head Division, for their cooperation with this research, particularly Jonathan Ross and Chester Clark.

AUTHOR BIOGRAPHIES

MARK A. TREADWELL is a student in the Department of Mechanical Engineering at the University of Maryland, College Park. He is a member of INFORMS and ASME. His e-mail address is <mtread@umd.edu>.

JEFFREY W. HERRMANN is an associate professor at the University of Maryland, College Park, where he holds a joint appointment with the Department of Mechanical Engineering and the Institute for Systems Research. He is the director of the Computer Integrated Manufacturing Laboratory. He is a member of INFORMS, ASME, IIE, SME, and ASEE. He received his Ph.D. in industrial and systems engineering from the University of Florida. His e-mail address is <jwh2@umd.edu>.