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

OPERATOR-PACED ASSEMBLY LINE SIMULATION

.	Marvin S. Seppanen

Productive Systems
2225 Garvin Heights Road
Winona, MN 55987, U.S.A

ABSTRACT

This paper describes an Arena-based operator-paced assembly line simulation model. Besides the normal flow of assembly units between work cells, the model considers both the movement of operators between cells and intermittent duties, such as equipment repair and material stocking. Simulation results are presented for a 4-cell test case.

1 INTRODUCTION

Discrete event simulation has been widely used to model assembly lines. In particular the automotive industry has long used simulation and more recently animation tools to assess the impact of various assembly line designs (Gujarathi et al. 2004) (Roser et al. 2003). For the most part, past models have concentrated on the mechanical aspects of assembly line design and largely ignored the human or operator component (Baines et al. 2003). The Arena model presented in this paper augments the standard assembly line model to include intermittent operator duties. These duties may consume a significant portion of the operator's available time and restrict the operator's ability to perform the primary assembly operations.

The Arena model presented in this paper is specifically designed to incorporate the indirect or intermittent duties the assembly line operators are periodically required to perform. The model is a generalized version originally developed for a client using operator-paced assembly lines to produce a variety of building supply products. The model can be used to test strategies for reducing lost production caused by intermittent operator duties.

2 PROBLEM

Assembly lines are typically modeled as flow of assembly units or entities through a series of workstations or cells. This type of flow inherently fits the network modeling approach of simulation tools such as Arena. Figure 1 illustrates the assembly system modeled for this paper.

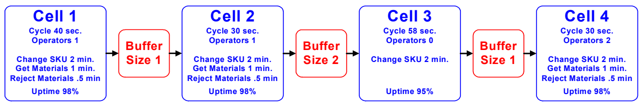


Figure 1: Assembly Line Flow Diagram

The assembly line flow diagram illustrated in Figure 1 is comprised of four cells sequentially linked with three buffers having capacities of 1, 2, and 1 units respectively. Cells 1 and 2 each require one operator to function, Cell 3 is automatic and needs no operator, and Cell 4 requires two operators to function. This assembly line is designed to be operated with two to four operators. One goal of the simulation is to determine the impact that the number of operators has on throughput. In addition to the basic tasks, the operators are periodically required to obtain material to support the assembly operations, repair failed equipment, inspect and reject defective material, and change the equipment setup when a new Stock Keeping Unit (SKU) or product type is being assembled.

3 MODEL DESIGN

The operator-paced assembly line simulation model is developed using the standard technique developed by the author (Seppanen 1995, Seppanen 2000, Seppanen et al. 2005) for linking Arena to Excel using Visual Basic for Applications (VBA). All model data are contained in a single Excel Parameters workbook. Separate worksheets contain each type of model information. This method of simulation model data transfer has becomes a more common method of simulation model design (Qiao et al. 2003).

This paper does not include detailed description of the VBA code; however, both the Arena 8.01 model and associated Parameters workbook are available at, `<http://factory.engr.stthomas.edu/simulation/operator/>`. Figure 2 illustrates the flow of information between Excel, Arena, and Arena 3-D Player. File Transfer.bat is generated by the VBA code to automate the saving of model results to a subfolder. The model can also generate file Model.pbf to be used as input to the

Arena 3D Player program for animating the simulation results in a 3D format.

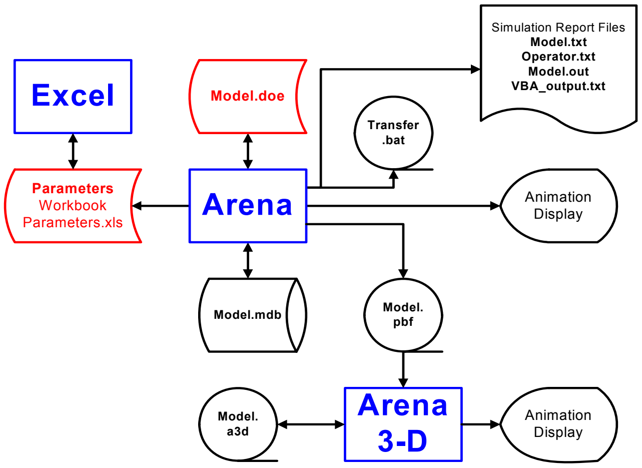


Figure 2: Model Environment

3.1 Entities

The model uses Arena entities {Entity.Assembly.Unit} to simulate the flow of assembly units through the assembly process. A second entity type {Entity.Failure} is used to control the cell operation and repair cycles. Each assembly unit entity carries attributes used to store the current assembly cell, the assigned operators, and product information.

3.2 Cells

Figure 3 illustrates the general Arena logic used to model assembly cell functions. Because Arena supports Sets, this logic can handle any number of cells by simply extending the set definitions, see Figure 5.

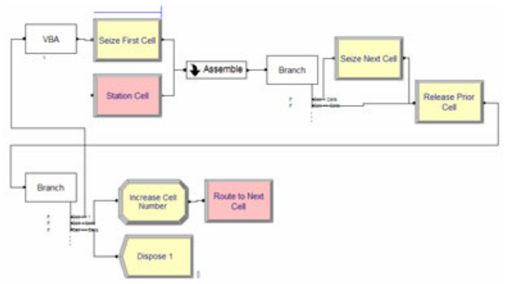
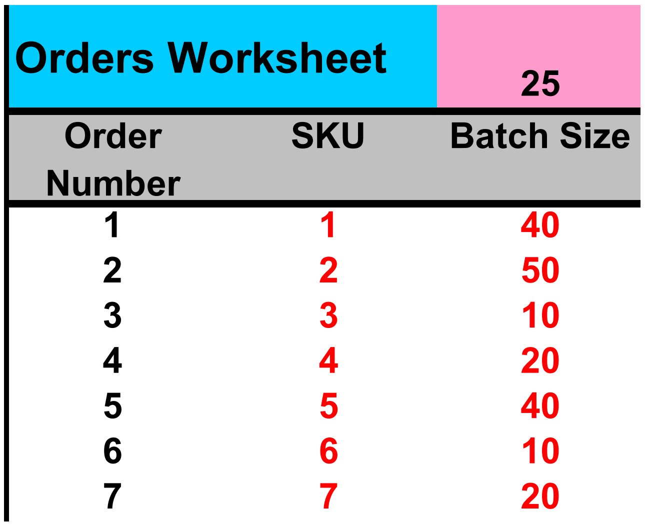


Figure 3: Cell Model logic

The first assembly unit entity enters the VBA block from an external source. The VBA block invokes VBA code that sets the assembly unit entity's SKU, order number, and serial number attributes. These calculations are driven by the data in worksheet Orders which specifies a sequence of SKU's to be produced and the batch size for each order, see Figure 4. The data set is automatically repeated when the data in worksheet Orders has been exhausted. The VBA code also sets the entity's Cell attribute to an initial value of 1.



Orders Worksheet		25
Order Number	SKU	Batch Size
1	1	40
2	2	50
3	3	10
4	4	20
5	5	40
6	6	10
7	7	20

Figure 4: Worksheet Orders

Next the cell resource R.Cell.1 or R.Cell (1) is seized. The Station Cell module defines the set of stations {S.Cell = S.Cell.1, S.Cell.2, S.Cell.3, S.Cell.4}. The Assemble submodel, which will be discussed later, controls the operator interactions and simulates the assembly process.

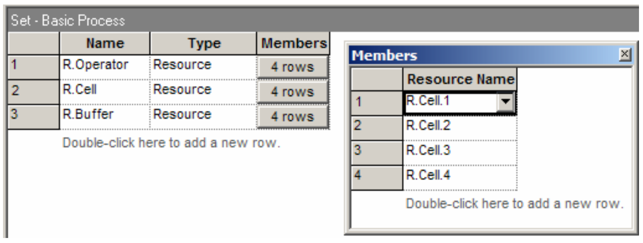
The first branch block controls the downstream processing of the assembly unit entity. If the current Cell number is less than the total number of modeled cells {Cell < Cells} the next cell resource {R.Cell (Cell + 1)} is seized. Next the current cell resource {R.Cell (Cell)} is released.

The second branch block determines if the current cell is the first {Cell == 1}. If so, a copy of the original entity is sent to the VBA block to start the processing of a new assembly unit entity. This methodology assures a continuous flow of incoming assembly units without the excess queuing of entities when the assembly process cannot keep pace with a predetermined arrival rate.

If the current cell is less than the maximum {Cell < Cells}, the original assembly unit entity has its Cell attribute increased by one and is routed to the next assembly cell. If the current cell is equal to the maximum number of cells {Cell == Cells}, the original assembly unit entity is disposed or removed from the simulation model.

The Arena modules illustrated in Figure 3 are typically augmented with additional modules to record production statistics and resource status. This modeling approach does not explicitly model a conveyor or other type of material transportation system. Rather it prevents the movement of assembly unit entities until the next assembly cell resource is available. If this is not possible, the cell resource status is recorded as blocked. The routing time to the next assembly cell represents the transit time between the appropriate cells.

Figure 5 illustrates the definition of resource set {R.Cell = R.Cell.1, R.Cell.2, R.Cell.3, R.Cell.4}. This set could be expanded to support a model with additional cells by increasing the number of resources: R.Cell.5, R.Cell.6, etc. Similar expansion of queue and station sets would also be required.



Set - Basic Process

	Name	Type	Members
1	R.Operator	Resource	4 rows
2	R.Cell	Resource	4 rows
3	R.Buffer	Resource	4 rows

Double-click here to add a new row.

Members

	Resource Name
1	R.Cell.1
2	R.Cell.2
3	R.Cell.3
4	R.Cell.4

Double-click here to add a new row.

Figure 5: Cell Resource Set Definition

3.3 Buffers

The Arena model illustrated in Figure 3 assumes a fixed succession of assembly cells. No buffer or float space is permitted between the assembly cells. The model developed for this paper has been made more general by optionally permitting buffer spaces between the assembly cells. Figure 6 illustrates the modules used to replace the three upper-right modules in Figure 3 to incorporate a buffer between successive assembly cells.

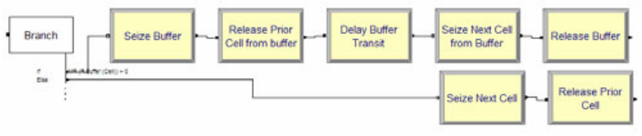
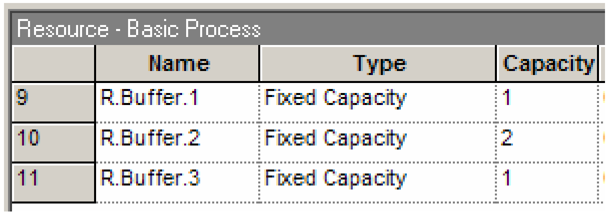


Figure 6: Buffer Model Logic

The branch block tests for the presence of buffer following the cell assembly {MR (Buffer (Cell)) > 0}. If the expression is false, no buffer is present and the normal processing illustrated in Figure 3 is executed. If the expression is true, the assembly unit entity first seizes a unit of buffer resource {R.Buffer (Cell)}. When buffer space has been seized, the prior assembly cell resource {R.Cell (Cell)} is released. The assembly unit entity is then held or delayed for the minimum buffer transit time. After this delay, the assembly unit entity seizes the next assembly cell resource {R.Cell (Cell + 1)}. When the next assembly cell resource has been seized, the buffer resource {R.Buffer (Cell)} is released. Figure 7 illustrates the buffer resource definition for The Arena model.

The buffer sizes and minimum buffer transit times are specified on the CycleTimes worksheet of Parameters workbook rather than being hard coded into the model as illustrated in Figure 7. Note, Arena permits the definition of resources with a capacity of zero. The logic illustrated in Figure 6 skips all null buffers {MR (Buffer (Cell)) == 0}.



Resource - Basic Process

	Name	Type	Capacity
9	R.Buffer.1	Fixed Capacity	1
10	R.Buffer.2	Fixed Capacity	2
11	R.Buffer.3	Fixed Capacity	1

Figure 7: Buffer Resource Definitions

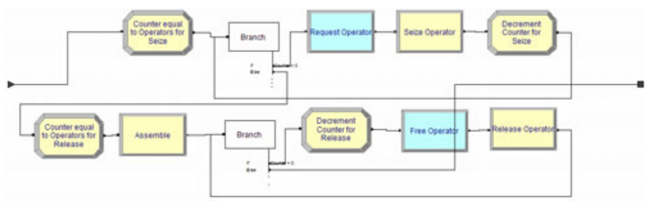
3.4 Operators

Assembly line operators are typically modeled as a resource such as R.Operator, with a capacity of possibly 2, 3, or 4 for the four-cell case study. Arena Sets permit a more detailed modeling construct by defining a set {R.Operator = R.Operator.1, R.Operator.2, R.Operator.3, R.Operator.4}. This set structure permits the tracking of individual operators. However, it does not provide an easy method to account for the movement of operators between assembly cells. Considerable operator movement between cells is to be expected if the four-cell assembly line were to be simulated with two or three operators. Omission of operator movement might significantly impact the model's validity.

Arena's transporter construct offers an alternative for modeling the mobility of assembly line operators. When a transporter is requested from a pool of identical transporters, one of several options can be used to control the assignment. We elected to take the closest available transporter. Arena then calculates the movement time from the transporter's current location to the requesting assembly cell location. This calculation requires the model to contain distance information for the various possible transporter paths and the associated travel speeds.

The transporter construct would be sufficient to model assembly line operators if detailed usage information and operator scheduling were not desired. Because detailed operator usage statistics and operator break periods were desired, operators were modeled using transporters and resources in parallel.

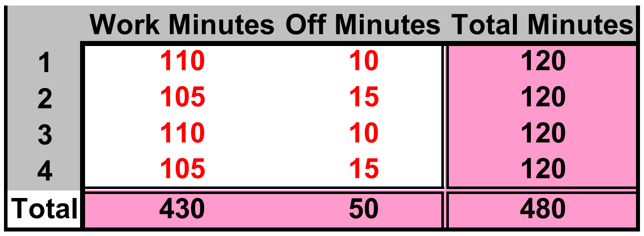
Figure 8 illustrates the logic detail for the Assemble submodel introduced in Figure 3. When an operator is required for an assembly process, the Operator transporter is first requested from the pool of available Operators {T.Operator}. The selected transporter number is recorded in attribute Temp. After the selected transporter has moved to the requesting assembly cell location, the corresponding operator resource {R.Operator (Temp)} is seized. A similar logic is used to release the operator after the completion of the assembly process.


Counter equal to Operators for Seize
Branch
Request Operator
Seize Operator
Decrement Counter for Seize
Counter equal to Operators for Release
Assemble
Branch
Decrement Counter for Release
Free Operator
Release Operator
Figure 8: Assemble Submodel Logic

The logic presented in Figure 8 includes additional complexity because more than one operator may be required to perform the assembly operation, see Cell 4 in Figure 1. Therefore an additional attribute, Counter, is used to track the number of operators remaining to be seized or released. The branch blocks control the looping until the required number of operators have been seized or released. This same branch block automatically handles the situation when no operator is required; see Cell 3 in Figure 1. Some additional bookkeeping is required to track the specific operators being used.

3.5 Operator Breaks

Unless assembly line operators use a “tag relief” system, the assembly line is halted to accommodate operator breaks. The length of the work and break periods are specified in the Schedule worksheet, see Figure 9. Fifty minutes of a 480-minute shift are break or off minutes. This leaves 430 minutes available for work duties. The Arena model uses VBA code to remove operators from service for the duration of their scheduled breaks.



	Work Minutes	Off Minutes	Total Minutes
1	110	10	120
2	105	15	120
3	110	10	120
4	105	15	120
Total	430	50	480

Figure 9: Worksheet Schedule

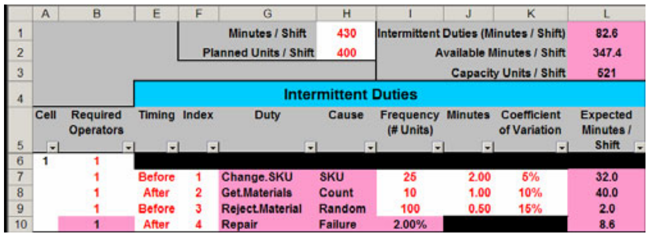
Arena does not permit the number of transporters to be altered during the model execution. Rather the number of operator resources is reduced to zero for the duration of each break period. The logic illustrated in Figure 8 first requests the transporter and then seizes the resource. During the break periods when the operator resource capacity has been reduced to zero, the assembly unit entity waits in the seize queue until the operator returns from break. This records the transporter as busy even though it is waiting for the operator resource to return from a break.

3.6 Intermittent Operator Duties

The logic described in the Operators section outlined the process used to model assembly operators. Similar logic is also employed to model intermittent operator duties that might be required either immediately before the assembly operation or shortly after its completion. The Arena model incorporates four types of intermittent operator duties: Change.SKU, Get.Materials, Reject.Material, and Repair. These intermittent operator duties are defined in worksheet Duties.

The specifics of the intermittent operator duties are input to the model using the CycleTimes worksheet. Figure 10 illustrates the intermittent duties associated with Cell 1. This worksheet also specifies the frequency and duration of the intermittent duties. Cell 1 has four intermittent operator duties:

1. Change.SKU occurs before starting the assembly processing whenever the assembly unit entity’s SKU changes or on an average of every 25 units with duration of 2.0 minutes.
2. Get.Materials occurs after completing the assembly processing every 10 units with duration of 1.0 minute.
3. Reject.Material occurs before starting the assembly processing at random about every 100 units or with a probability of 0.01 with duration of 0.5 minutes.
4. Repair occurs whenever an equipment failure takes place. This type of failure takes place 2% of the time based on information in the Failures worksheet, see Figure 11.



	A	B	E	F	G	H	I	J	K	L
1					Minutes / Shift	430	Intermittent Duties (Minutes / Shift)			82.6
2					Planned Units / Shift	400	Available Minutes / Shift			347.4
3							Capacity Units / Shift			521
4			Intermittent Duties							
5	Cell	Required Operators	Timing	Index	Duty	Cause	Frequency (# Units)	Minutes	Coefficient of Variation	Expected Minutes / Shift
6	1	1								
7		1	Before	1	Change.SKU	SKU	25	2.00	5%	32.0
8		1	After	2	Get.Materials	Count	10	1.00	10%	40.0
9		1	Before	3	Reject.Material	Random	100	0.50	15%	2.0
10		1	After	4	Repair	Failure	2.00%			8.6

Figure 10: Cell 1 Intermittent Operator Duties

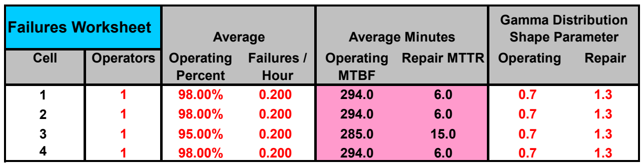
All simulated intermittent operator duty times are randomly generated using the normal distribution and the specified coefficient of variation.

Worksheet Duties also contains static analysis to estimate the total time spent on intermittent operator duties. In the case of Cell 1 about 82.6 minutes per shift will be spent on intermittent operator duties assuming a planned production rate of 400 assembly units per shift. This leaves 430 – 82.6 = 347.4 minutes for assembly operations. Assuming an average assembly time of 40 seconds, Cell 1 should be able to produce, 347.4 * 60 / 40 = 521 units per shift.

The logic for implementing the intermittent operator duties is very similar to that used for the assembly operations, see Figure 8. Submodels DutiesBefore and DutiesAfter define the required Arena logic. VBA code is used to determine when intermittent operator duties are to be performed and their durations. Note that the number of operators required can vary for each intermittent operator duty.

3.7 Equipment Failures and Repair

Each assembly cell is assumed to contain equipment that is subject to random failure. The specific data for modeling these failures and their subsequent repairs are contained in worksheet Failures, see Figure 11. This model uses the gamma distribution shape parameters suggested by Law (1990 and 2000). Cells 1, 2, and 4 are expected to be available for operation 98% of the time. Their operating periods are generated from a gamma distribution with a mean of 294 minutes and shape parameter of 0.7. Failed or repair periods are generated from a gamma distribution with a mean of six minutes and shape parameter of 1.3. The repair intermittent operator duty may be delayed while waiting for an operator. The repair of all four cells requires the use of one operator.



Failures Worksheet		Average		Average Minutes		Gamma Distribution Shape Parameter	
Cell	Operators	Operating Percent	Failures / Hour	Operating MTBF	Repair MTTR	Operating	Repair
1	1	98.00%	0.200	294.0	6.0	0.7	1.3
2	1	98.00%	0.200	294.0	6.0	0.7	1.3
3	1	95.00%	0.200	285.0	15.0	0.7	1.3
4	1	98.00%	0.200	294.0	6.0	0.7	1.3

Figure 11: Worksheet Failures

4 SIMULATION RESULTS

Figure 12 illustrates the Arena model execution with three operators. At the time of the screen capture, one operator is conducting a Change.SKU operation at Cell 1 and the other two operators are doing the assembly operation at Cell 4. A single assembly unit entity is present at Cell 2 and two units at Cell 3. Figure 13 illustrates the Arena 3D Player animation also with three operators.

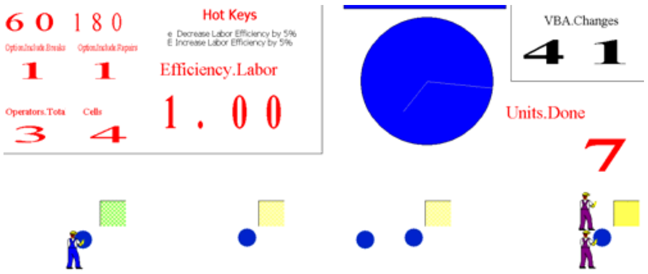


Figure 12: Arena Animation Display

Arena permits the collection of detailed state statistics for resources. Figure 14 illustrates the state statistics for resource R.Cell.2 while Figure 15 illustrates the state statistics for resource R.Operator.1.

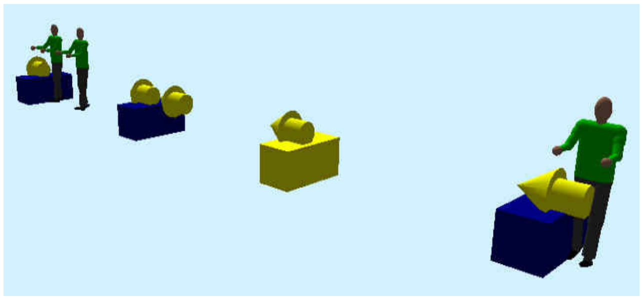
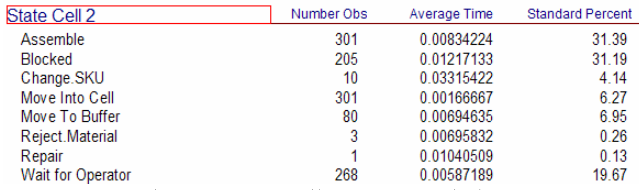
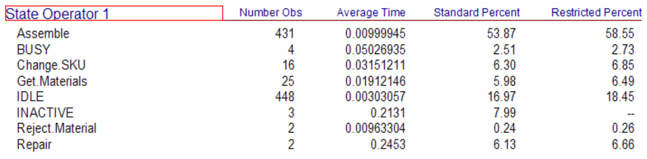


Figure 13: Arena 3D Player Animation Display



State Cell 2	Number Obs	Average Time	Standard Percent
Assemble	301	0.00834224	31.39
Blocked	205	0.01217133	31.19
Change.SKU	10	0.03315422	4.14
Move Into Cell	301	0.00166667	6.27
Move To Buffer	80	0.00694635	6.95
Reject.Material	3	0.00695832	0.26
Repair	1	0.01040509	0.13
Wait for Operator	268	0.00587189	19.67

Figure 14: R.Cell.2 State Statistics



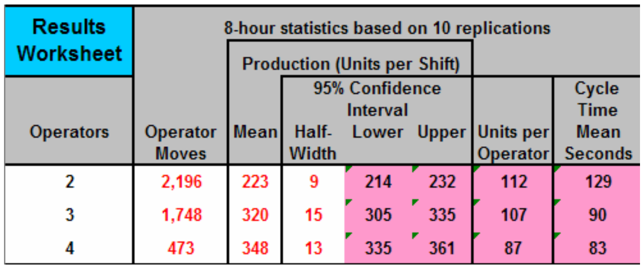
State Operator 1	Number Obs	Average Time	Standard Percent	Restricted Percent
Assemble	431	0.00999945	53.87	58.55
BUSY	4	0.05026935	2.51	2.73
Change.SKU	16	0.03151211	6.30	6.85
Get.Materials	25	0.01912146	5.98	6.49
IDLE	448	0.00303057	16.97	18.45
INACTIVE	3	0.2131	7.99	--
Reject.Material	2	0.00963304	0.24	0.26
Repair	2	0.2453	6.13	6.66

Figure 15: R.Operator.1 State Statistics

Cell 2 spent about 31% of the shift in both the Assemble and Blocked states. A total of 301 assembly units were processed. Of the remaining part of the shift the most significant portion, 20% is spent waiting for an operator. Increasing the number of operators from three to four could reduce that statistic. The remainder of the shift is spent moving the assembly unit entity into the cell or from the cell into the buffer or on intermittent operator duties.

The breakdown of time spent on intermittent operator duties can be seen in Figure 15. Operator 1 spent 54% of the 8-hour shift on assembly duties or 59% of the available 430 minutes during the shift. It should be noted that Operator 1 is idle 17% of the shift, while from Figure 14 Cell 2 is waiting for an operator a similar fraction of shift. Unfortunately these lost times do not overlap and therefore cannot be put to practical use.

The Arena model does not provide a direct means to determine how much of the operator time is spent traveling to the next work location. Travel time would be included in the idle time statistics of Figure 15. The model did collect statistics for the number of times an operator moved to a new work location. That data is summarized for three different operator levels in Figure 16. The number of operator moves is highly dependent on the number of assigned operators. Notice that even when four operators were assigned slightly more than one operator move is simulated for each assembly unit produced. This result seems counterintuitive but could be explained as follows. Intermittent operator duties performed after the assembly operation does not prevent the assembly cell from starting the processing of the next assembly unit entity. Therefore, Cell 1 and Cell 2 may at times be simultaneously using two operators while Cell 4 might at times be using three operators. Each use of an extra operator involves two operator moves.



Results Worksheet	8-hour statistics based on 10 replications						
		Production (Units per Shift)					
			95% Confidence Interval				Cycle Time
Operators	Operator Moves	Mean	Half-Width	Lower	Upper	Units per Operator	Mean Seconds
2	2,196	223	9	214	232	112	129
3	1,748	320	15	305	335	107	90
4	473	348	13	335	361	87	83

Figure 16: Simulation Results Summary

Figure 16 also indicates the number of assembly units produced per shift in terms of mean and its 95% confidence interval. These statistics were based on 10 replications of eight hours or one shift each. Notice that while the mean production per shift dynamically increased with the number of assigned operators, the production per operator slightly decreased as more operators were used. This presents an interesting optimization problem for further study. The mean effective cycle time is also calculated as function of the number of operators.

5 CONCLUSION

The model presented in this paper demonstrates the feasibility of including intermittent operator duties in addition to the standard assembly line paced duties. Because all data required to drive this model are contained in a single Parameter workbook, it is very easy to test alternative operating scenarios. The model can be expanded without altering the logic changes to accommodate any number of cells and/or operators. Such change would involve expansion of the model sets, variable array sizes, and animation structure.

The model described in this paper assumes that all operators are located in a single pool. This assumption may not be valid for an assembly line with a large number of cells and/or operators. Further the assumption that all operators are equally skilled and therefore are interchangeable. While such as assumption might not be true in practice, it would require considerable effort and addition data to accurately incorporate into a simulation model.

REFERENCES

Baines, T., L. Hadfield, S. Mason, and J. Ladbrook. 2003. Using empirical evidence of variations in worker performance to extend the capabilities of discrete event simulations in manufacturing. *In Proceedings of the 2003 Winter Simulation Conference*, ed. S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice. 1210-1216. Piscataway, New Jersey: IEEE.

Gujarathi, N. S., R. M. Ogale, and T. Gupta. 2004. Production capacity analysis of a shock absorber assembly line using simulation. *In Proceedings of the 2004 Winter Simulation Conference*, ed. R. G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, 1213-1217. Piscataway, New Jersey: IEEE.

Law, A. M. 1990-1. Models of random machine downtimes for simulation, Industrial Engineering, 22 No. 8, August, 58-59.

Law, A. M. 1990-2. Models of random machine downtimes for simulation, Industrial Engineering, 22 No. 9, September, 22-23.

Law, A. M., and W. D. Kelton. 2000. *Simulation modeling and analysis*, 3rd Edition, McGraw Hill.

Kelton, W. D., D. T. Sturrock, and R. P. Sadowski. 2003. *Simulation with Arena*, 3rd Edition, McGraw Hill.

Roser, C., M. Nakano, and M. Tanaka. 2003. Buffer allocation model based on a single simulation. *In Proceedings of the 2003 Winter Simulation Conference*. ed. . S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice. 123-1246. Piscataway, New Jersey: IEEE.

Seppanen, M. S., 1995. Developing industrial strength simulation models. *In Proceedings of the 1995 Winter Simulation Conference*. ed. C. Alexopoulos, K. Kang, W. R. Lilegdon, and D. Goldsman. 936-939. Piscataway, New Jersey: IEEE.

Seppanen, M, S., 2000. Developing industrial strength simulation models using Visual Basic for Applications (VBA). *In Proceedings of the 2000 Winter Simulation Conference*, ed J.A. Joines, R. R. Baron, K. Kang and P. A. Fishwick. 77-82. Piscataway, New Jersey: IEEE.

Seppanen, M. S., S. Kumar, and C. Chandra. 2005. *Process analysis and improvement: tools and techniques*, 1st Edition, McGraw Hill.

Qiao, G., F. Riddick, and C. McLean. 2003. Data drive design and simulation system based on XML. *In Proceedings of the 2003 Winter Simulation Conference*. ed. . S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice. 1143-1148. Piscataway, New Jersey: IEEE.

AUTHOR BIOGRAPHY

MARVIN S. SEPPANEN, Ph.D., P.E. His company, Productive Systems, an independent Industrial Engineering consulting firm specializing in simulation modeling and capacity analysis of manufacturing systems. Marvin holds BME, MSIE, and Ph.D. (Operations Research) degrees from the University of Minnesota. Before starting Productive Systems he was an Associate Professor of Industrial Engineering at General Motors Institute and The University of Alabama. He is a Registered Professional Engineer; Senior Member and chapter officer, Institute of Industrial

Engineers; Member, The Society for Computer Simulation; Member, Society of Manufacturing Engineers; and has been certified at the Fellow Level by the American Production and Inventory Control Society. He teaches simulation using Arena to Manufacturing Engineering students at the University of St. Thomas in St. Paul, Minnesota. His e-mail address is `<seppanen@hbci.com>` and his Web address is `<www.stthomas.edu/engineering/faculty/seppanen.asp>`.