Modeling Stochastic Inventory Policy with Simulation

János BENKŐ
Department of Material Handling and Logistics, Institute of Engineering Management

Keeping an inventory (stock of goods) for the future sale or use is very common in business. Retail firms, wholesalers, manufacturing companies have a stock of goods on hand. Management faces a lot of challenging issues involving difficult decisions which can be company-oriented and customer-oriented issues. For example how does such a firm decide upon its inventory policy which is an important part of supply chains. In studying this problem, modelers typically focus on the following key performance metrics, which eventually can be translated to monetary measures: customer service levels (the rate of totally satisfied customer demands), average inventory levels and backorder levels, rate and quantity of lost sales and inventory cost.

1. Introduction

The importance of inventory as a part of supply chains comes from the fact that they constitute a large segment of economic activities. To improve their activities, to gain competitive advantage companies are interested in effective and efficient inventory operations that meet customer expectations. Therefore the supply chain management (SCM) or logistics tasks to produce and distribute products in the right quantities to the right locations at the right time, while keeping costs down and customer service levels up. In essence, SCM aims to solve these tasks by searching for good trade-offs between system costs and customer satisfaction.

SCM faces a lot of challenging issues involving difficult decisions which can be company-oriented and customer-oriented issues. In this study we deal with customer-oriented issues. For example, a customer's demand may not be fully satisfied from stock on hand and the shortage may be backordered or the sale may be lost. Either way, it is desirable to balance the holding cost which depends on the magnitude of inventory against the cost of not fully satisfying customer demand.

In studying such issues, modelers typically focus on the following key performance metrics, which eventually can be translated to monetary measures: (1) customer service levels (the rate of totally satisfied customer demands), (2) average inventory levels and backorder levels, (3) rate and quantity of lost sales, (4) inventory cost.

To achieve good performance, supply chains employ inventory control policies that regulate the orders to replenish stocks. Inventories can be reviewed continuously or periodically. The well-known inventory policies used in industry: (t,Q), (t,S), (s,Q), (s,S), where t is the cycle time, Q is the order quantity, s is the reorder level and S is target level. In the first two cases the reorder point is determined by the cycle time. In the other cases replenishment of the inventory starts when the inventory level reaches the reorder level.

Typically, when an order is performed by a producer or retailer, there is a lag in time (called lead time) after which the order is received. The lead-time demand is the magnitude of demand that arises during lead time, and is typically random and therefore can result in stock-outs. Consequently, to decrease the uncertainty in lead-time demand, companies constitute safety stock, which is extra inventory stocked to maintain good customer service levels. Companies may also use to place new orders before previous ones are received. Therefore ordering decisions are typically made based on inventory positions rather than inventory levels, where the

Inventory position = Inventory level + Inventory on order – Backorders.

Companies to get feasible inventory management use scientific inventory policies which based on mathematical models. These models describe the behavior of the inventory system. Inventory models are classified according to whether the demand is known (deterministic) or whether it is random variable having a known probability distribution (stochastic). In the last case the chance of the exact solution is very limited therefore we use simulation.

2. Problem statement

Application of simulation will be demonstrated with a very simple production-inventory system consisting of a production facility which supplies a warehouse with one type of product. This generic model illustrates how an inventory control policy regulates the flow of product between production and inventory facilities.

Figure 1 depicts a schematic diagram of the system. The raw material storage feeds the production process, and finished product units are stored in the warehouse. Customers arrive at the warehouse with product requests (demands), and if a request cannot be fully satisfied by on-hand inventory, the unsatisfied portion represents lost sale.

The following assumptions are made:

There is always sufficient raw material in storage, so the process never starves.

Product processing is carried out in lots of 5 units, and finished lots are placed in the warehouse. Lot processing time is uniformly distributed between 10 and 20 minutes.

The production process experiences random failures that may occur at any point in time. Times between failures are exponentially distributed with a mean of 200 minutes, while repair times are normally distributed, with a mean of 70 minutes and a standard deviation of 30 minutes.

The warehouse operations implement the (s, S) inventory control policy with target level S=500 units and reorder point s=150 units.

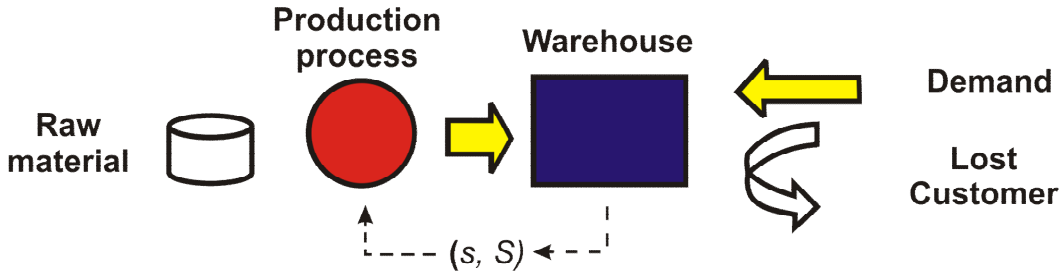

Figure 1. A generic production-inventory system

Customers arrive with interarrival times distributed as uniform with 3 to 7 hours, and individual demand quantities are also distributed uniformly between 50 and 100 units. On customer arrival, the inventory is immediately checked. If there is sufficient stock on hand, that demand is promptly satisfied. Otherwise, the unsatisfied portion of the demand is lost.

The initial inventory is 250, so the production process is initially idle.

Ordering cost has two components: K constant setup cost (regardless of the order quantity) is 10000 Ft/setup and c unit price 100 Ft/unit. The specific holding cost h is 2 Ft/unit/hour and the specific shortage cost p is 8 Ft/unit/hour.

We are interested in to determine the following performance parameters:

Production process utilization.

Downtime probability of the production facility.

Average inventory level at the warehouse.

Average demand level.

Percentage of customers whose demand is not completely satisfied.

Average lost demand quantity, given that it is not completely satisfied.

Average total operating (inventory) cost per hour.

The production-inventory problem described previously, although fairly elaborate, is still a simplification of real-life supply-chain systems. However, the applied method can be extended to more realistic problems, including multiple types of products, multiple production stages, production setups, and so on.

3. Arena model

Having studied the problem statement, we now construct the Arena model of the production-inventory system. Figure 2 shows our Arena model which is composed of two segments:

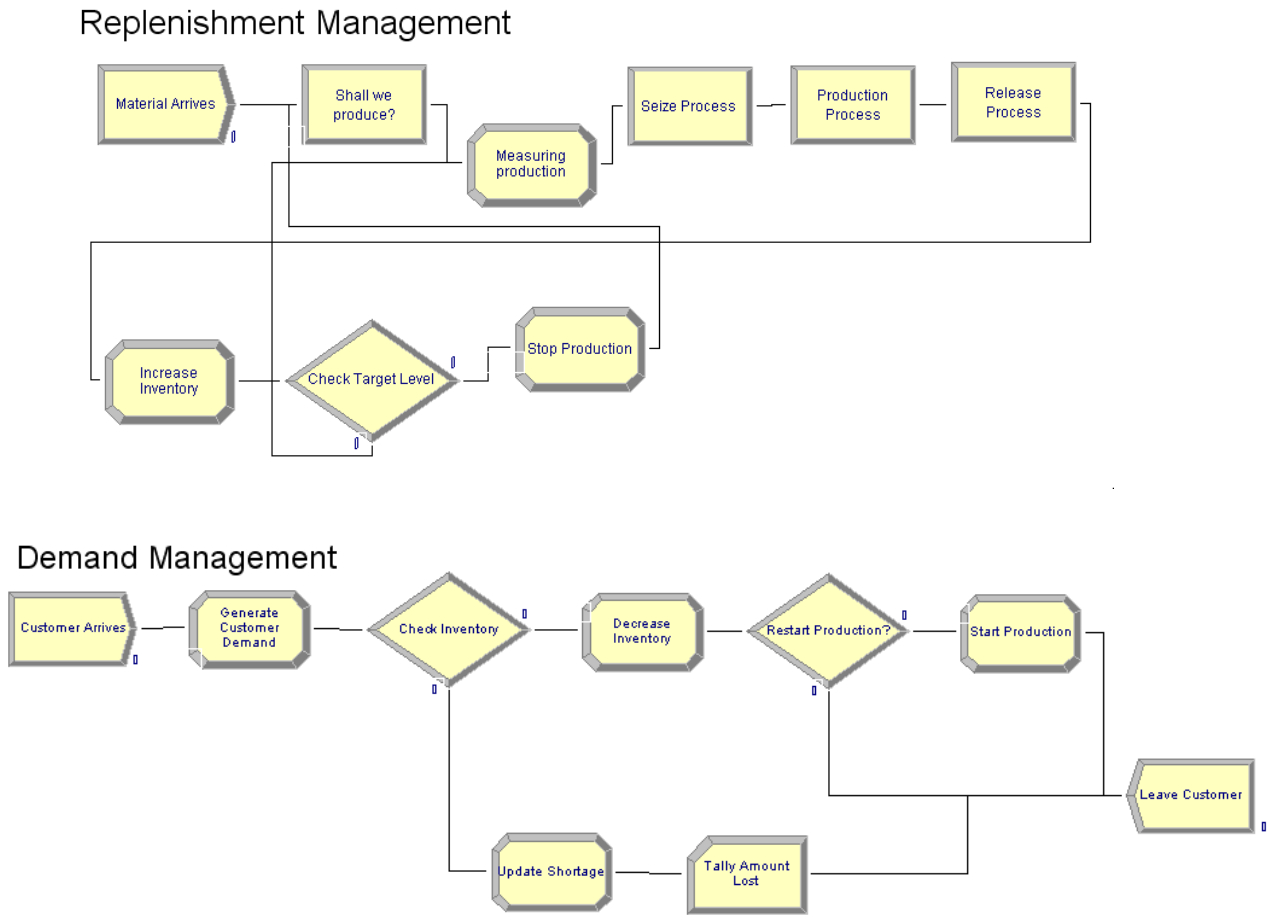

Figure 2. Arena model of the production-inventory system

Replenishment management segment. This segment keeps track of product unit entities. Entity definitions can be inspected and edited in the spreadsheet view of the **Entity** module from the **Basic Process** template panel. In this part of the model, the production process takes a unit of raw material from its queue, processes it as a batch of 5, and adds the finished lot to the warehouse inventory, represented by the variable *Inventory*. Thus, when a lot entity completes processing, *Inventory* is incremented by 5, and the simulation logic branches to one of two outcomes as follows: (1) If *Inventory* up-crosses the target level

(variable *Target Stock*), then production stops until the reorder point (variable *Reorder Point*) is down-crossed again. (2) Processing of a new batch starts immediately when the reorder point is down-crossed. (Note that we always have sufficient raw material, so the production process never starves).

Demand management segment. This segment generates customers and their demands and adjusts variable Inventory upon customer arrival. It monitors the value of Inventory, and triggers resumption of suspended production when the reorder point is down-crossed. It also keeps track of lost demand (customers whose demand is not fully satisfied).

In addition, input and output data logic is embed in the two segments above. This logic consists of input/output modules (variables, resources, statistics, etc.) that set input variables, compute statistics, and generate summary reports. In the following sections, we proceed to examine the Arena model logic of Figure 2. in some detail.

Replenishment management segment

We start with the replenishment management portion of the logic. The **Create** module, called *Raw Material*, generates product units for the production operation. A **Hold** module, called *Shall We Produce?*, serves to control the start and stop of the operation by feeding product units into a sequence of **Seize**, **Delay**, and **Release** modules (called *Seize Process*, *Production Process*, and *Release Process*), all drawn from the **Advanced Process** template panel. The actual processing takes place at the **Delay** module, called *Production Process*, where the process time of a batch is specified as *Unif*(10, 20) minutes.

The **Create** module *Raw Material* was initialized by populating it with a single product entity (at time 0), and therefore the interarrival time is irrelevant. Furthermore, by setting the *Max Arrivals* field to 1, the **Create** module will deactivate itself thereafter. The product entity circulates in the model repeatedly, with each cycle representing a production cycle.

The circulating product entity then proceeds to the **Hold** module, called *Shall We Produce?*, to test if production is turned on. The state of production (*Off*=0 or *On*=1) is maintained in the variable *Production*, which is initially set to 0. Recall that the **Hold** module performs a gating function on an entity by scanning for the truth or false of a logical condition. If the condition (in our case, *Production* = 1) is *true*, then the product entity proceeds to the next module. Otherwise, it waits in queue *Shall We Produce?.Queue* until the condition becomes *true* before being allowed to proceed. The circulating product entity then proceeds to queue *Production Queue* in the *Seize Process* module and seizes the resource called *Production facilities* immediately.

In the queue of the **Seize** module called *Production Queue* product entities await their turn for one unit of resource (*Production facilities*) to perform the process operation at module *Production Process*. Once the resource becomes available, it is seized by the highest-ranking product (in this case, rank 1). While the current process is in progress, the **Seize** module blocks any additional product entities from entering it. On service completion, the **Release** module (which never denies entry) releases one unit of resource *Production Process*.

A spreadsheet view of the **Resource** module with an entry for *Production facilities* is shown at the bottom of Figure 3. with its *Failures field* (associating resources to failures). All failure/repair data (uptimes and downtimes) are specified in the **Failure** spreadsheet module, as illustrated in Figure 4.

The inventory level at the warehouse is maintained by the variable *Inventory*, initially set to 250. Whenever the circulating product entity (batch of five units) enters the **Assign** module, called *Increase Inventory*, it adds a batch of 5 finished units (variable *Batch Size*) to the warehouse inventory by just incrementing *Inventory* by the batch size.

The circulating product entity then goes to the **Decide** module, called *Check Target Level*, whose dialog box is shown in Figure 5. Here, the circulating product entity tests whether the inventory target level has been up-crossed. The test can result in two outcomes:

If the inventory target level has been up-crossed, then the circulating product entity moves on to the following **Assign** module, called *Stop Production*, and sets *Production*=0 to signal that production is suspended.

Otherwise, the circulating product entity repeats productions.

Either way, the circulating product entity has completed its task through the system and would normally be disposed of (at a **Dispose** module). However, since the *Production Process* module is never starved and no delay is incurred since its departure from the *Production Process* module, we can "recycle" the circulating product entity by always sending it back to the *Production Process* module to play the role of a new arrival. This modeling device is logically equivalent to disposing of a product entity and creating a new one. However, it is computationally more efficient, since it saves us this extra computational effort, so that the simulation will run faster.

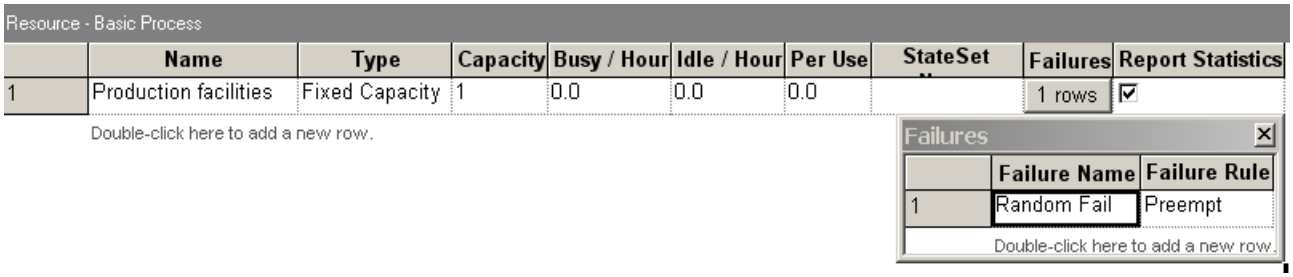

Figure 3. Dialog spreadsheet of the **Resource** module with its **Failures** dialog spreadsheet

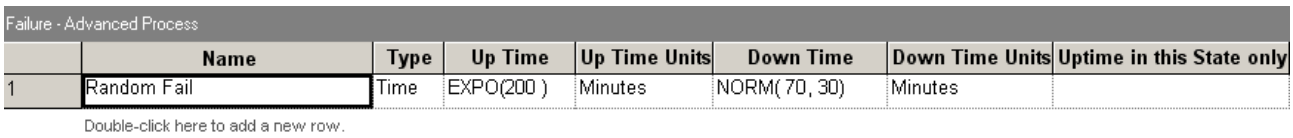

Figure 4. Dialog spreadsheet of the **Failure** module

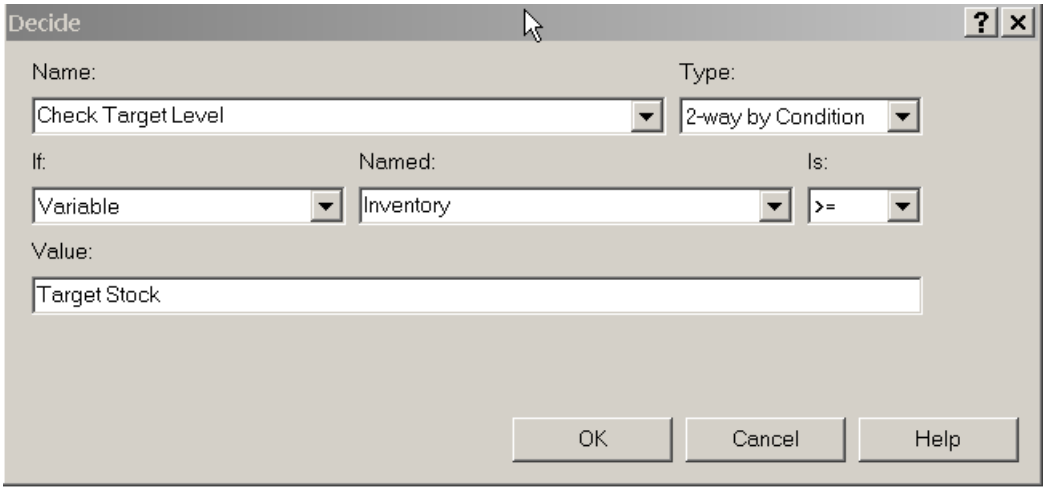

Figure 5. Dialog box of the **Decide** module *Check Target Level*.

Demand management segment

The source of customer demand at the warehouse is the **Create** module, called *Customer Arrives*. The arrival pattern of customers is specified to be random with interarrival time distribution *Unif*(3, 7). Each arrival is an entity of type *Customer* (customer entity) with its private

set of attributes (e.g. a demand quantity attribute, called *Demand*). On arrival, a customer entity first enters the **Assign** module, called *Generate Customer Demand,* where its *Demand* attribute is assigned a random value from the *Unif*(50, 100) distribution. The customer entity then proceeds to the **Decide** module called *Check Inventory* to test whether the warehouse has sufficient inventory on hand to satisfy its demand. The test can result in two outcomes:

If the value of variable *Inventory* is greater or equal to the value of attribute *Demand,* then the current demand can be satisfied and the customer entity takes the *True* exit to the **Assign** module, called *Decrease Inventory*, where it decrements the inventory by the demand amount. It next proceeds to the **Decide** module, called *Restart Production?*, to test whether the *Reorder Point* variable has just been down-crossed. If it has, the customer entity proceeds to the **Assign** module, called *Start Production,* to set *Production*=1, which would promptly release the circulating product entity currently detained in the **Hold** module *Shall We Produce?*, effectively resuming the production process. Either way, the customer entity proceeds to be disposed of at the **Dispose** module, called *Leave Customer.*

If the value of variable *Inventory* is strictly smaller than the value of attribute *Demand*, then the current demand is either partially satisfied or not at all. Either way, the customer entity proceeds to the **Assign** module, called *Update Shortage,* where it sets the *Inventory* variable to 0. It also updates the variable *Lost Customer*, which keeps track of the number of customers to-date whose demand could not be fully satisfied (*Lost Customer=Lost Customer*+1), and the attribute called *Amount Lost*, which keeps track of the demand lost by the current customer (*Amount Lost =Demand–Inventory*). The customer entity next enters the **Record** module, called *Tally Amount Lost,* to tally the lost quantity per customer whose demand was not fully satisfied. Finally, the customer entity proceeds to be disposed of at module *Leave Customer.*

Variable - Basic Process

	Name	Rows	Columns	Clear Option	Initial Values	Report Statistics
1	Inventory			System	1 rows	☐
2	Production			System	1 rows	☐
3	Total Customer			System	0 rows	☐
4	Target Stock			System	1 rows	☐
5	Reorder Point			System	1 rows	☐
6	Batch Size			System	1 rows	☐
7	Lost Customer			System	0 rows	☐
8	Specific production cost Ft per Unit			System	1 rows	☐
9	Specific holding cost Ft per Unit per Unit time			System	1 rows	☐
10	Specific shortage cost Ft per Unit per Unit time			System	1 rows	☐
11	Number of Setup			System	0 rows	☐
12	Amount Production			System	0 rows	☐
13	Beginning of Cycle			System	1 rows	☐
14	Length of Cycle			System	0 rows	☐
15	Total Amount Lost			System	0 rows	☐
16	Total Demand			System	0 rows	☐
17	Setup cost Ft per number of setup			System	1 rows	☐
18	Demand			System	0 rows	☐

Double-click here to add a new row.

Figure 6. Dialog spreadsheet of the **Variable** module.

The user-defined variables involved in the model can be set or inspected by clicking on the **Variable** module of the **Basic Process** template panel. This yields the spreadsheet view of the module, as exemplified in Figure 6.

Statistics collection

The **Statistic** module is used to define nonstandard additional statistics (it is called *User Specified Statistic*) that are to be collected during the simulation and also to specify output data files.

Figure 7. displays the dialog spreadsheet of the **Statistic** module for the production-inventory model. It includes 4 Time-Persistent (DSTAT) type statistic, called *Average Inventory Level in Unit*, for the *Inventory* variable, *Average Demand Level in Unit*, for the *Demand* variable, *Average Cycle Length in Unit time*, for the *Length of Cycle* variable, and *Production On*, for the expression *Production*=1. The Time-Persistent type statistical outputs consist of the average value, 95% confidence interval, and minimal and maximal values of the variables ever observed during the replication. For example, the statistical output for *Production*=1 is the rate of time when this expression is true, or that is the probability that the production process is set to active. (Keep in mind, however, that the production process may experience downtimes.) Recall that this method of time averaging expressions can be used to estimate any probability of interest, including joint probabilities.

Statistic - Advanced Process

	Name	Type	Expression	Report Label
1	Average Inventory Level in Unit	Time-Persistent	Inventory	Average Inventory Level in Unit
2	Process State	Frequency		Process State
3	Production On	Time-Persistent	Production==1	Production On
4	Lost Rate in Customer per Total Customer	Output	Lost Customer/Total Customer	Lost Rate in Customer per Total
5	Average Demand in Unit per Customer	Output	Total Demand/Total Customer	Average Demand in Unit per Customer
6	Production Cost in Ft per Unit time	Output	Amount Production * Specific production cost Ft per Unit/TFIN	Production Cost in Ft per Unit time
7	Holding Cost in Ft per Unit time	Time-Persistent	Specific holding cost Ft per Unit per Unit time * Inventory	Holding Cost in Ft per Unit time
8	Shortage Cost in Ft per Unit time	Output	Total Amount Lost * Specific shortage cost Ft per Unit per Unit	Shortage Cost in Ft per Unit time
9	Setup Cost in Ft per Unit time	Output	Setup cost Ft per number of setup*Number of Setup/TFIN	Setup Cost in Ft per Unit time
10	Total Inventory Cost in Ft per Unit time	Output	OVALUE(Setup Cost in Ft per Unit time)+OVALUE(Production	Total Inventory Cost in Ft per Unit time
11	Average Cycle Length in Unit time	Time-Persistent	Length of Cycle	Average Cycle Length in Unit time
12	Average Demand Level in Unit	Time-Persistent	Demand	Average Demand Level in Unit

Double-click here to add a new row.

Figure 7. Dialog spreadsheet of the **Statistic** module.

Figure 7 also contains a *Frequency* type statistic, called *Process States*, which estimates the state probabilities of the production process, namely, the probabilities that the production process is busy or down.

The first from the six Output type statistics in Figure 7 used to calculate the lost rate of customers that suffered some lost demand. The corresponding Expression field there indicates that when the replication terminates, the variable *Lost Customer* is divided by the variable *Total Customer* to yield the requisite rate. The other Output type statistics calculate the elements of inventory cost and the total inventory cost.

4. Result, discussion

Figure 8 displays reports of the results of a simulation run of length 20,000 hours (more than 2 years).

The results in Figure 8 give quite instructive information about performance of the production-inventory model.

The expression *Amount Lost Per Customer* in the Tally section indicates that the average lost demand per customer who experienced some loss was about 22 units. The Half Width column estimates the half-length of respective confidence intervals at the 95% confidence level.

The *Average Inventory Level in Unit* statistic in the Time Persistent section shows that occasionally the inventory up-crossed its target level. However, this happened only rarely,

because the *Production* variable in the Time Persistent section assumed the value *On*=1 about 98.54% of the time. The average inventory level is 160.54 units. It means that often the inventory level was lower, than the reorder level, *s*=150. It seems in Figure 9. which shows the changing of the inventory and demand level in time.

14:43:43 **User Specified** december 1, 2010

Production Inventory Replications: 1

Replication 1 Start Time: 0,00 Stop Time: 16 666,67 Time Units: Hours

Tally

Expression	Average	Half Width	Minimum	Maximum
Amount Lost Per Custumer	22.0868	1,51424	0.04322332	75.3473

Time Persistent

Time Persistent	Average	Half Width	Minimum	Maximum
Average Cycle Length in Unit time	1,037.18	(Insufficient)	0	3,267.90
Average Demand Level in Pc	75.1384		0	99.98
Average Inventoty Level in Pc	159.20	20,08239	0	504.58
Holding Cost in Ft per Unit time	318.40	40,16477	0	1,009.17
Production On	0.9854	(Insufficient)	0	1.0000

Output

Output	Value
Average Demand in Unit per Customer	75.2366
Lost Rate Customer per Total Customer	0.1214
Production Cost in Ft per Unit time	1,455.78
Setup Cost in Ft per Unit time	6.6000
Shortage Cost in Ft per Unit time	4.3043
Total Inventory Cost in Ft per Unit time	1,785.09

Figure 8. Simulation results for the production-inventory model

The Output statistic *Lost Rate in Customer per Total Customer* in the Output section reveals that 12.02% of the customers had their demand either partially satisfied or not satisfied at all.

The *Total Inventory Cost in Ft per Unit time* also in the Output statistic is 1790.84 Ft/hour. Its components: the *Setup Cost in Ft per Unit time* is 6 Ft/hour, the *Production Cost in Ft per Unit time* is 1455.78 Ft/hour, the *Holding Cost in Ft per Unit time* is 321.09 Ft/hour, and the *Shortage Cost in Ft per Unit time* is 4.27 Ft/hour.

As we mentioned before, goal of SMC is to achieve economic balance between system costs and customer satisfaction. It known that simulation itself does not solve this problem directly, however, we can vary the input parameters of the system and we can observe the changes of the outputs. For example, we vary the value of *Reorder point* between 50 and 150 units, because we would like to decrease the inventory cost and to improve the service level (probability that the demand of an arriving customer is fully satisfied) and the utilization of the production facility. Table 1 consists of the simulation results. Increasing *Reorder point* until 100

units results decreasing costs, improving facility utilization and the lost rate does not change significantly. *Reorder point*=100 units can be considered as an optimum, because above 100 units the cost increases and the facility utilization marginally falls down. This simple analysis leads us to conclude that a significant cost decrease can be achieved by choosing the appropriate reorder point.

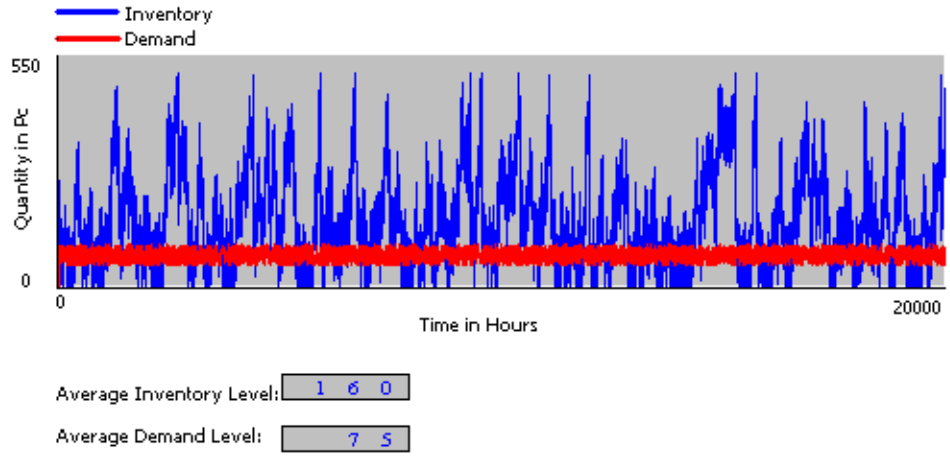

Figure 9. *Inventory* and *Demand* level as function of the simulation time

Table 1.

Cost analysis of the production-inventory system

Reorder Point in Unit	Holding cost in Ft/hour	Shortage cost in Ft/hour	Setup Cost in Ft/hour	Total Inventory Cost in Ft/hour	Lost Rate in %	Avg. Cycle Time in hour	Production on in %
50	349.87	3.21	6.0	1819,30	9.89	1314.49	98.25
75	288.09	4.20	4.0	1753,72	12.80	1532.36	98.91
100	296.68	3.75	3.5	1763,49	11.63	2477.92	99.02
125	303.74	4.35	4.0	1769,84	12.64	2261.50	98.94
150	321.09	4.27	6.0	1790,84	12.02	980.64	98.68

Much more information can be gained from the statistical output, considering the restricted extent of this study we could not strive for it. We encourage you to complete the analysis of the simulation results.

REFERENCES

1. **Altiok, T.:** Simulation Modeling and Analysis with Arena. Elsevier Inc. ISBN 13: 978-0-12-370523-5, 2007.

2. Arena Professional Reference Guide, Rockwell Software Inc., 2000.

3. **Bratley, P.,-Fox, B. L.-Schrage, L. E.**: A Guide to Simulation. Springer-Verlag, New York, NY, 1987.

4. **Benkő, J.**: Modeling Kanban Systemic Production with Arena Simulator. GépGyártás, Vol. XLIX, Nr. 4., 2009.

5. **Devore, L.**: Probability and Statistics for Engineering and the Sciences. Wadsworth Inc, Belmont, CA., 2003.

6. **Fishman, G. S.**: Principles of Discrete Event Simulation. John Wiley, 1978.

7. **Kelton, W. D., Sadowski R. P., Sturrock, D. T.**: Simulation with Arena. Mc Graw Hill Higher Education, ISBN 0-07-285694-7, 2004.

7. **Mitra, D., and I. Mitrani.** (1990): Analysis of a kanban discipline for cell coordination in production lines. I. Management Science 36: 1548-1 566.

Publikálva:

Mechanical Engineering Letters 2010, Szent István University, Gödöllő, 2010, 190-200 p.