Proceedings of the 2010 Winter Simulation Conference
B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, eds.

SIMULATING BACKLOG AND LOAD BUILDING PROCESSES IN A TWO-ECHELON INVENTORY SYSTEM

Manuel D. Rossetti

Department of Industrial Engineering
University of Arkansas
4207 Bell Engineering Center
Fayetteville, AR 72701 USA

Yisha Xiang

School of Business
Sun Yat-sen University
135 W. Xingang Road
Guangzhou, Guangdong 510275 China

ABSTRACT

In this paper, we discuss the design and use of an object-oriented framework for simulating a two-echelon inventory system. We present how the framework can be used to simulate the backorder and load building queues at the warehouse level of the system. In addition, we describe the modeling options for the backorder processing for replenishment orders sent to the warehouse. Filled orders must then be consolidated into loads for shipping to the retailer level. The framework is built on a Java Simulation Library (JSL) and permits easy modeling and execution of simulation models. A set of experiments is performed to illustrate how queueing disciplines for the backorder and load building queues effect the lead-time experienced at the retailer level. In addition, we summarize future research efforts to model complex supply chains.

1 INTRODUCTION

A supply chain can be considered as a network of locations (or facilities) and distribution options that operate to obtain raw materials, transform these materials to finished products, distribute these finished products to the customers depending upon customer demand requirements. This paper presents simulation models for a supply chain having multiple items stocked at multiple locations and involving transportation between the locations. This paper builds on previous efforts to develop an object-oriented simulation framework for simulating a supply chains with tree and general network structures. The focus of this paper is on types of supply chains that form a general class of inventory systems called multi-echelon inventory systems. Figure 1 illustrates the basic structure of a multi-echelon inventory system.

The management of individual stock keeping units (SKUs) within such a network is an important issue. A SKU is an individually managed item type, for which a quantity of inventory is stored and managed at a particular location. When there are more than two echelons, one location (e.g. a warehouse) supplies another location (e.g. a retail outlet). When there are more than two echelons, the context of warehouse-retail is less defined. Because the common notion is that these locations hold inventory, the term, inventory holding point (IHP), is used here to denote a location that holds multiple types of items.

It is important to note that the same item type may be held at multiple IHPs within the network. Each of these item-location combinations can be considered as a different SKU. Stock keeping units might follow different stocking policies resulting in decentralized control of the supply network. In addition, different transportation options are used to move materials between locations when replenishments are required. Determining the performance (e.g. fill rate, expected number of backorders, inventory cost, transportation cost, etc.) of each SKU at each location is extremely challenging due to the complex stochastic processes and material movement rules.

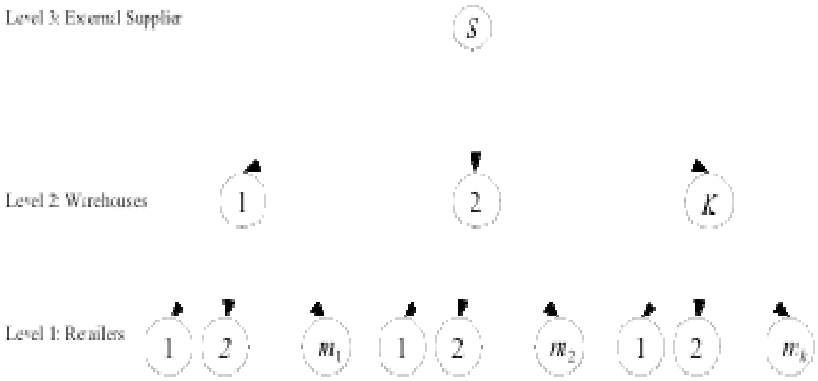

Figure 1: Multi-Echelon Inventory System

This paper represents a continuation of the work in Rossetti and Chan (2003), Rossetti et al. (2006), and Rossetti et al. (2007) and also presents new modeling concepts/examples. The intention of the paper is to illustrate how to model the backorder queue and the load building queues within a two-echelon inventory system. In addition, the paper illustrates the effect that different queueing disciplines can have on the performance of individual stock keeping units within the network. This should serve as a basis for researchers and practitioners that might be interested in building better analytical models of these processes.

The rest of this paper is structured as follows. The next section presents a brief overview of the literature in this area to give context for how this modeling fits into multi-echelon methods. Then, Section 3 examines the simulation modeling constructs used within the paper. This overviews the use of the Java Simulation Library (JSL) and the inventory package for modeling multi-echelon supply chains. In addition, details are provided on how to conceptualize backorder queue processing and load building processes. Section 4 discusses a set of experiments that illustrate the effect of the modeling and presents the results of the experiments. Finally, the last section describes some areas for future work.

2 BACKGROUND AND LITERATURE REVIEW

Overviews of inventory modeling can be found in Axsater (2006), Zipkin (2000), Muckstadt (2005), Tempelmeir (2006), and Silver et al. (1998). Multi-echelon inventory systems consist of two or more echelons where locations at upper echelons act as suppliers to locations at lower echelons. Figure 1 illustrates only a single external supplier that serves multiple warehouses, which support multiple retailers. In general, other support patterns (e.g. multiple suppliers, lateral shipments, etc.) are possible. Work on multi-echelon inventory systems has been an active area for decades (e.g. Clark and Scarf (1960), Sherbrooke (1968), Sherbrooke (1986),) Gümüs and Güneri (2007) indicate that simulation is the most used research technique in their review of over 92 references.

The dependency between echelons makes it hard to model each location without considering other locations. At the lowest echelon (retailer level), replenishment orders are a function of direct customer demand. The item's lead-time at any location is a random variable that is a function of the stocking and operating policies at the next higher echelon (Cohen et al. 1990). For example, at any location, the expected number of backorders during the lead-time is a function of the inventory control policies. Hence, a replenishment order placed by a location to a higher echelon might experience a random delay due to a stockout. This implies that the replenishment lead-time is a function of the waiting time experienced at the next higher echelon, which is, in turn, a function of its stocking policy. Thus, the modeling of any location depends on the stocking policy and operating processes of its supplying location at the next higher echelon. These and other relationships tie echelons together, which implies that the echelons must be modeled simultaneously in order to be able to adequately model their performance characteristics.

The coordination of inventory management and transportation management is very important for an efficient supply chain management, and has received more attention recently. Existing logistics literature

identifies three main types of temporal shipment consolidation routines (Higginson and Bookbinder, 1995): time-based policies, quantity-based policies, and the time-and-quantity policy. In a series of related papers, Çetinkaya and Lee (2000, 2006, 2008) analyze the impact of shipment consolidation on inventory control arising in the context of Vendor-managed Inventory (VMI) and highlight the significant cost savings that may be realized by integrated policies. Their analysis concentrates on simultaneously computing the optimal order quantity for inventory replenishment at the vendor and the optimal dispatching frequency/quantity for outbound shipments. Wang and Takakuwa (2006) develop a module-based method to analyze the production-distribution systems by using a discrete event simulation with ARENA. Both multistage multi-product inventory control and shipment consolidation are considered in their study.

A key modeling issue is whether or not an order can be divided during processing. Zipkin (2000) suggests that the primary service measure is the expected (total amount) backordered measured in units. However, when modeling the waiting time between levels, it may be better to conceptualize whole orders waiting. In reality, customers do not necessarily permit their orders to be divided and in practice dividing an order can be operationally unattractive due to the potential cost of processing the demands individually and the tracking convenience of keeping the customer order together. At the very least the cost of splitting the order should be considered. Figure 2 illustrates orders of different sizes waiting in the backorder queue. If customer orders can be divided, the model is essentially a (r, q) model since unit demand occurs and no undershoot of the reorder point may happen. As Zipkin (2000) indicates, the modeling of orders that are kept together can "lead to a far more complex model". It should be clear that if the order cannot be divided and units are not allocated to an order until it can be completely filled, then the waiting times of orders from different retailers might be significantly different (e.g. larger orders may have to wait longer than smaller orders).

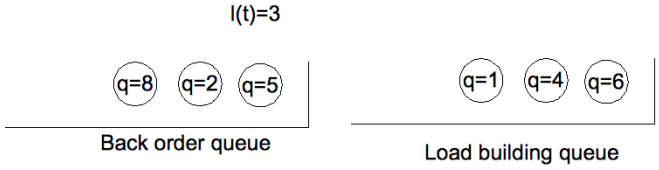

Figure 2: Batch Orders of Different Sizes Waiting for Filling and Load Building

Not dividing the orders should necessitate an analysis of the retailer's orders as different classes of customers in the queue, take into account the size of the order, and possibly utilize batch queuing models. Since the expected waiting time and the variance of the waiting times are incorporated into the lead-time for the retailers, assuming that the retailers all experience the same wait is inappropriate when orders cannot be divided. Matheus and Gelders (2000) analyze an (r, q) model under compound Poisson demand and use the splitting property of the Poisson process to split the arrivals into sub-streams; however, this modeling is predicated on permitting the order to be divided. See also Chen and Zheng (1994) and Chiang and Chiang (1996).

The problem of interest in this study concerns how queueing disciplines for the backorder and load building queues effect the lead-time experienced at the retail level in a two-echelon inventory system. The literature on the class of this problem is scarce. Zheng and Zipkin (1989) develop a queuing model where two products competing for a single production facility, and examine the impact of different processing rules such as the longest queue first policy. Iyer (2001) consider a supply chain consisting of a single production facility that serves many retail locations. They examine the impact of moving from a First Come First Served (FCFS) production rule for orders arriving at the production facility to a rule in which a non-preemptive priority (PR) is provided to orders from retail locations with higher demand uncertainty. Their analysis suggests that tailoring lead times to product demand characteristics may decrease system inventory costs.

After an order has been filled, in practice, it proceeds to a staging area where the orders can be shipped. In much of the research literature, this load building is ignored or incorporated into the transport

time between levels. This assumes that the items are processed independently. However, the size and timing of the loads has been shown to be an important element in the control of these systems. For example, the q units of an item outbound for a retailer may be accumulated in a queue (see Figure 2). The items may wait until a full truckload is built, either by cube or weight, or they may wait in the queue until a time limit is reached. Thus, the items interact with each other through the load-building process. See Rossetti and Liu (2009) or Magableh et al. (2005) for examples of how these systems are simulated. In addition, the warehouse might decide to round the order up or down in order to take advantage of transportation economics. Mason et al. (2009)) indicate that such rounding rules can have a significant effect on inventory levels. A review of the multi-echelon inventory literature indicated a lack of the explicit modeling of the load building processes. Since load-building rules can add variability into the delay based on order size and can make the performance estimation process non-separable by item type, a better understanding of the effect on performance modeling and optimization of explicitly modeling the load building process is needed. The following section describes simulation models that can assist with this understanding.

3 INVENTORY MODELING AND THE JSL

Rossetti et al. (2007) present an object-oriented framework for simulating multi-echelon inventory systems. A framework is a set of reusable classes that allow for modeling within a particular domain. This paper builds on the multi-echelon inventory framework by adding additional modeling capabilities for simulating the backordering and load building processes. The models presented here depend on the Java Simulation Library.

3.1 Overview of the JSL

The Java Simulation Library (JSL) is an open-source simulation library that facilitates discrete-event simulation modeling in the Java programming language. A description of the JSL is provided within Rossetti (2008). The purpose of the JSL is to support education and research within simulation. The JSL has packages that support random number generation, statistical collection, basic reporting, and discrete-event simulation modeling. The development of a simulation model is based on sub-classing the ModelElement class that provides the primary recurring actions within a simulation and event scheduling and handling. The user adds developed model elements to an instance of Model and then executes the simulation. The design of the JSL provides a framework for the testing of simulation artifacts through a well-defined class and interface hierarchy. The structure of the JSL permits the easy switching of various components within a simulation, the event calendar, random number generator, statistics, etc.

The library is divided into Java packages (calendar, modeling, observers, testing, and utilities). As necessary, these packages may contain sub-packages, which implement various aspects of the library.

- calendar – The calendar package implements classes that provide event calendar processing
- modeling – The modeling package is the heart of the JSL. In the modeling package, supporting model elements such as queues, resources, variables, commands, etc. are implemented.
- observers – The observers package provides support classes for observing model elements during the simulation run. The JSL is designed to allow each model element to have observers attached to it thereby implementing the classic Observer pattern. Observers can be used to collect statistics, write output to files, display model element state, etc.
- utilities – The utilities package provides support classes that are used by the JSL. The random, reporting, and statistic sub-packages are in this package.

Using these packages, additional packages were developed to model inventory and transportation options within supply chains.

3.2 Overview of Multi-Echelon Inventory Modeling in the JSL

Rossetti et al. (2006) and Rossetti et al. (2007) describe in detail the capabilities with the JSL for simulating multi-echelon supply chains. Two Java packages within the JSL form the basis for the modeling framework presented in this paper:

- *supplychain.inventorylayer* – Provides simulation constructs for modeling the use and location of inventory within general supply chain structures.
- *supplychain.transportlayer* – Provides abstract and concrete classes that model the movement of loads between inventory holding points.

The *inventorylayer* package consists of 33 interfaces, 7 abstract classes and 51 concrete classes of which Table 1 lists the key classes with respect to this paper. For simplicity in the presentation, objects that implement an interface or that are instances of a particular class are referred to by the lower case nouns related to the class or interface names whenever the context is clear. For example, item type and demand filler refer to instances of ItemType and DemandFillerIfc, respectively. Inventory models using the inventory layer package consist of objects that send demand (implementing the DemandSenderIfc) and those that fill the demands (implementing the DemandFillerIfc).

ItemType represents items within the inventory system. An item type has a weight, a cube, and an external lead time. Demand represents a request for inventory for a particular item type. A demand has attributes referencing the item type associated with the request, the sender of the request, the filler of the inventory items, the carrier of the items, the amount of the request, and other customer requirements. This may include whether or not backlogging or partial filling is permitted. A DemandGenerator creates end customer demands and sends the demand to a filler. Inventory represents units of a particular item type that can fill demand requests. Thus, Inventory acts as a demand filler by implementing the DemandFillerIfc interface. Inventory represents the state of the stock keeping unit and tracks of the amount of on-hand, backlogged, on order, etc. Every Inventory class is associated with an inventory policy. InventoryPolicyAbstract acts as an abstract base class the represents a rule, policy, or strategy that governs the reordering behavior. A variety of different inventory policies, such as continuous and periodic review policies, are available. This paper focuses on reorder point, reorder quantity policies (i.e. (r,Nq) policies). Statistics are tracked on the amount on hand, backlogged, the ready-rate (i.e. the proportion of time that net inventory is positive) as well as many other state variables by SKU and by IHP.

Whenever a demand is filled by the inventory or whenever a replenishment arrives at the inventory, the inventory position is updated. The checking of the inventory position and the subsequent responses are handled by the inventory policy classes. If the inventory is immediately filled from stock on hand, then its "first fill rate" is updated. That is, the probability that the entire order can be satisfied directly from the shelf. This distinction is important since the warehouse experiences lumpy demand (based on the reorder quantity) that will not be (in general) governed by a Poisson process. The first fill rate and the ready rate can be significantly different in this situation. If the end customer demand cannot be directly satisfied, it will be backlogged.

If a replenishment order is triggered it is sent to the designated filler for the inventory, in this case an instance of the InventoryHoldingPoint class that represents the external supplier or the warehouse. An InventoryHoldingPoint contains many instances of the Inventory class to represent the SKUs. The IHP acting as the warehouse sends its replenishment orders to the external supplier, which uses a simple lead time delay to represent production and transport to the warehouse. This is implemented by a LeadTimeDemandFiller.

Table 1: Important Classes in *inventorylayer* and *transportlayer* packages

Class Name	Description
ItemType	Describes items and products
Demand	Represents a request for inventory
DemandState	Represents the status of the request for the inventory
InventoryPolicyAbstract	Abstraction of the inventory policy
BackLogPolicyAbstract	Abstraction of the backlog policy
DemandFillerIfc	Represents objects that fill requests for inventory
DemandFillerFinderIfc	Represents objects that find a filler to fill the demand
DemandSenderIfc	Represents objects that send requests for inventory
DemandGenerator	Encapsulates the demand creation
Inventory	Something that holds inventory
InventoryHoldingPoint	IHP, Represents locations that stock inventory items (sub class from InventoryHolderAbstract)
LeadTimeDemandFiller	Represents locations that produce items and fills demand requests after a time delay
DemandCarrierIfc	Represents a transportation option to move demands between IHPs
TimeBasedDemandCarrier	Transports demands (as loads) from filler to customer
DemandLoad	Represents a set of filled demands that requires transport
DemandLoadBuilder	Represents the thing that builds loads for shipment to a customer
DemandLoadFormingRuleIfc	Represents a rule to be used for forming loads
ScheduledLoadBuilderAbstract	A base class for load building based on a schedule
TimeBasedLoadCarrier	Represents a transportation option to move loads between IHPs

The DemandCarrierIfc, knows how to ensure that a filled demand is delivered. Classes that allow time-based transport between locations have been implemented based on the DemandCarrierIfc interface. The time to transport from the external supplier may depend on the location of the warehouse (IHP). If the time for the transport depends upon the link between the supplier and the customer, then a TimeBasedDemandCarrier is used to model the transport time. When demands are consolidated into load, then a TimeBasedLoadCarrier is used to represent the transport. This requires that the demands waiting for shipment are consolidated into loads. Both the backorder process and the load forming process are important aspects of the delay experienced at the warehouse.

3.3 Modeling Backorder Queues and Load Building Processes

The demand undergoes a series of state changes, described in detail in Rossetti et al. (2007). A demand has the following states (in preparation, negotiating, sent, rejected, received, cancelled, in process, filled, backlogged, shipped, and delivered). From these state changes, the time spent in each state can be captured. In this paper, the time spent backlogged, the time spent before shipping, and the transport time will be examined.

A backlog policy can be associated with the Inventory class. An abstract base class, BackLogPolicyAbstract, represents the different rules or behaviors that can be used to backlog demands for inventory and to fill backlogs associated with inventory. A backlog policy knows how to backlog the demand (causing it to wait in some manner) and how to fill demands that are waiting after a previously requested replenishment order arrives. BackLogPolicyAbstract permits any mechanism the user requires to hold the waiting demands. The default mechanism is a single queue that processes demands in a first come first served manner. Statistics can be captured to represent the number of demands backlogged, the total amount backlogged as well as time spent waiting in the backlog queue.

As previously mentioned, if a demand has an amount greater than 1 unit, there can be a number of different ways to process waiting demands. For example, while in the backlogged state, if the demand

permits partial filling, then it can be given increments of the amount requested until it becomes filled. Thus, first come first served may not be applicable in this case. For simplicity, this paper investigates the backlog queue using a priority mechanism based on the amount demanded. Demands that have less units demanded are placed at the front of the queue. Ties are handled based on first come first served. Since the demand from the retailers is lumpy, this will give retailers with smaller reorder quantities higher priority. The effect of this priority is explored in the experiments.

After demands are filled, they must be shipped to their respective customers. Because transport can be expensive, consolidating demands into loads destined for a single location is necessary. There are a number of ways consolidation can be accomplished. In practice, the goal is to utilize full-truck-load shipping as much as possible. To represent the queue to hold the demands waiting for shipment and the method for forming a load, the DemandLoadBuilder class can be used. It has default rules for processing filled demand waiting for transport and also allows the user supply their own load forming rules via the DemandLoadFormingRuleIfc. Load building strategies can be conceptualized as follows:

- By quantity. A load is formed when the quantity of orders accumulated in the buffer surpasses the minimum quantity for an economic shipment. Here quantity can denote the number, weight or cube of the ordered items, or any combination of limits for these attributes. For example, a load building rule may require that either the total weight in the load should exceed a certain minimum level or the total cube should exceed a certain level.
- By Schedule. If the warehouse keeps a fixed schedule for shipments to retailers, then whenever a scheduled shipment time is reached, a load is built out of whatever is in the buffer and the load is then shipped immediately. If the load is small then economies of transport may not be reached; however, the time that a load will arrive at the retailer can be better controlled. If the amount waiting in the load building queue is larger than an economic shipment size (i.e. a full truck) then, items may have to wait until the next scheduled shipment.
- Mixed rule. Duplicate rules can be imposed on the load building to make the shipment both flexible and reliable. For example, the warehouse can utilize shipment schedules and if the buffer grows sizable enough to form an economic load, the load can still be formed and shipped. In this way, a compromise between customer waiting and economies of transport can be investigated.

In this paper, each item type has a weight. A DemandLoadBuilder instance is created for each retailer to which the warehouse ships items. Rules can be specialized for each retailer. Demands wait in the load forming queue until a weight criteria is met. This assumes that trucks are loaded by weight in this example. Each load builder has a minimum weight criteria and a maximum weight criteria. Once the minimum weight criteria is met a load can be formed. The maximum criteria ensures that the load does not exceed this weight. This is necessary because the amount demanded varies by item type and by destination. In addition, if a schedule is used in conjunction with the weight criteria, the maximum ensures that whatever load is formed does not exceed the truck's capacity. After a load is built, shipment transportation starts immediately. The carrier takes the load to the destination and returns empty.

4 EXPERIMENTAL ANALYSIS AND RESULTS

The experimental investigation is based on the analysis of a base case network that has one warehouse and six retailers. Each retailer carries a same set of four item types but faces different customer demand rates. The weights of Item Types 1 and 2 are drawn from a Triangle distribution (5, 15, 25). The weights of Item Type 3 and 4 are drawn from Triangle distribution (30, 40, 50). Lead times from external supplier to warehouse are exponentially distributed. The mean of lead time for Item Types 1 and 3 are drawn from a uniform distribution (4, 6), and uniform distribution (6, 8) for Item Types 2 and 4. Transportation time from warehouse to all retailers is constant, drawing from a uniform distribution (2, 4). The different combination of demand rate, weight, external lead time, and transport time allows the retailers to have significantly different characteristics, which is important in understanding their interactions.

In the base case (r, Nq) inventory policies are used without considering load building. To better analyze the effect of different queue disciplines for backlog queues and let different retailers compete with each other, different order quantities for the same item type are specified across the retailers. The methods described in Axaster (2000) was used to find the reorder point subject to a fill rate constraint. The method described in Tempelmeier (2006) was used to aggregate orders from retailers into demands seen by warehouse. Then, by trial and error, acceptable inventory policies for the network were established. This results in some stock out probabilities at the warehouse being less than 20%, and for the retailers less than 40%. To incorporate load building into the base case, both the lower and upper weight limits for one load need to be determined. The intention here is to find the case where load building time is not a negligible part of the total lead time. Again, this was done by trial and error. The base case was run with load building for 10 replications with run length equal to 3650 days. Parameters of the model are summarized in Tables 2-4.

Table 2: Base Case Item Type and Warehouse Parameters

Item Type	External Lead Time	Weight	Reorder Point	Order Quantity (Q)
1	5.0437	15.22	5	30
2	6.853	14.24	80	120
3	4.9901	39.95	10	50
4	6.9417	39.7	30	50

Table 3: Retailer Base Case SKU Parameters

Retailer	A				B				C			
Item Type	1	2	3	4	1	2	3	4	1	2	3	4
Demand Rate	1.48	4.48	1.46	4.46	1.88	4.88	1.05	4.05	1.52	4.52	1.26	4.26
Reorder Point	5	15	5	25	6	15	2	20	6	15	5	25
Order Quantity	10	15	8	25	15	20	5	20	8	25	10	25
Retailer	D				E				F			
Item Type	1	2	3	4	1	2	3	4	1	2	3	4
Demand Rate	1.25	4.25	4.48	1.48	1.23	4.23	4.60	1.60	1.61	4.61	4.20	1.20
Reorder Point	5	10	25	5	5	12	22	5	2	12	20	12
Order Quantity	12	32	20	14	6	14	28	8	18	24	18	0.5

Table 4: Base Case Load Building Parameters

Load Forming	Min Load Weight	Max Load Weight
Retailer A	700	1000
Retailer B	1000	1200
Retailer C	1200	1500
Retailer D	1500	1800
Retailer E	1800	2000
Retailer F	2000	2500

Table 5: Base Case and Three Cases Results

Retailer	Item Type	Base Case			Case 1			Case 2			Case 3		
		Time In Backlog Q	Time In Load Building Q	Total Lead Time	Time In Backlog Q	Time In Load Building Q	Total Lead Time	Time In Backlog Q	Time In Load Building Q	Total Lead Time	Time In Backlog Q	Time In Load Building Q	Total Lead Time
A	1	3.02	2.08	7.90	1.66	2.13	6.59	3.02	3.16	8.98	1.66	0.00	4.46
	2	1.76	1.72	6.27	1.48	1.68	5.96	1.76	2.08	6.63	1.48	0.00	4.28
	3	2.00	1.60	6.41	0.83	1.73	5.36	2.00	1.83	6.63	0.83	0.00	3.63
	4	4.54	1.01	8.36	6.86	0.95	10.61	4.54	0.00	7.34	6.86	5.61	15.27
B	1	3.75	1.89	8.02	3.60	1.81	7.79	3.75	2.91	9.04	3.60	1.12	7.11
	2	2.00	1.82	6.20	1.74	1.72	5.84	2.00	1.90	6.28	1.74	1.01	5.14
	3	1.31	2.06	5.75	0.55	2.02	4.95	1.31	4.59	8.28	0.55	1.99	4.92
	4	4.18	1.40	7.96	1.58	1.48	5.44	4.18	0.47	7.02	1.58	2.05	6.00
C	1	2.57	2.43	8.19	1.31	2.58	7.07	2.57	5.10	10.86	1.31	2.59	7.08
	2	2.10	1.52	6.81	2.02	1.61	6.82	2.10	3.12	8.41	2.02	1.42	6.63
	3	2.43	2.12	7.74	1.05	2.22	6.46	2.43	1.83	7.45	1.05	1.79	6.04
	4	4.57	1.54	9.30	6.88	1.53	11.60	4.57	0.63	8.39	6.88	1.72	11.78
D	1	3.41	2.22	8.73	2.37	2.16	7.63	3.41	3.54	10.05	2.37	2.38	7.85
	2	2.19	2.98	8.27	3.50	2.74	9.34	2.19	5.17	10.46	3.50	2.24	8.84
	3	3.48	2.43	9.01	1.76	2.30	7.16	3.48	1.30	7.88	1.76	2.52	7.38
	4	3.66	2.49	9.25	1.30	2.56	6.96	3.66	1.77	8.53	1.30	1.65	6.05
E	1	1.91	2.70	7.13	0.99	2.98	6.49	1.91	8.56	12.99	0.99	2.89	6.40
	2	1.67	2.30	6.50	1.41	2.56	6.49	1.67	4.20	8.39	1.41	2.45	6.38
	3	4.02	2.85	9.39	8.81	2.76	14.09	4.03	1.14	7.68	8.81	3.18	14.51
	4	2.23	2.34	7.10	0.77	2.74	6.03	2.23	2.63	7.38	0.77	2.27	5.55
F	1	4.14	3.07	9.63	8.70	2.96	14.09	4.14	4.53	11.10	8.70	2.96	14.09
	2	2.10	3.06	7.59	1.82	3.06	7.32	2.10	3.41	7.94	1.82	3.06	7.32
	3	3.32	2.88	8.63	1.33	2.94	6.69	3.32	2.40	8.15	1.33	2.94	6.69
	4	3.36	2.96	8.74	1.16	2.97	6.57	3.36	2.94	8.73	1.16	2.97	6.57

To gain a better understanding of the backlogging processing and load-building rules, experiments were conducted to examine the impact of moving from FCFS process rules for both backlogging and load building queues to rules in which orders from retailers are ranked based on some given criteria. More specifically, priority is given to orders with the smallest amount needed for backlogging queues, and to orders with the smallest weight for the load building queues. By processing orders based on their priorities, our intension is to reduce the amount of time an order waits in backlogging and load building queues. This should reduce the total lead time. In addition to the base case, three other scenarios are considered to make the comparison: Case 1: ranked backlog queue and FCFS load building queue; Case 2: FCFS backlog queue and ranked load building queue; Case 3: ranked backlog queue and ranked load building queue. Each case is run with run length equal to 3650 days and for 1000 replications to get two decimal digits accuracy. Results of the base case with load building and Cases 1, 2, and 3 are summarized in Table 5.

From Table 5, it can be seen that, in Case 1 and 3, where priority rules are implemented for the backlogging queues, the waiting time of same item type with small order quantities in backlogging queue gets greatly reduced. See for example Item Type 1 at Retailers A, B, C, D, and E, etc.. The waiting time change for the load building queues for this case can be ignored.

It is expected that using a priority rule for load building queue will similarly reduce the waiting time of the orders with small weights from the same retailer, i.e. Item Types 1, 2 and 3 at Retailer A in Case 3. However, the effect of load building queue discipline is more complicated to examine. This is because the load building queue is by retailer and it is highly likely that only certain item type(s) from one retailer will wait in the load building queue. Thus, the priority rule may not have the expected impact. For example, if the lower weight limit of one load is set too high for some retailer, all orders might have to wait for the load to be formed. The sequence of arrival doesn't matter in this case. For instance, the sum of the weights of an order of Item Type 2 from Retailer C or D is the largest compared to the ones from the same retailer, but its waiting time at the load building queue still decreases. Our conjecture is that because Item Type 2 has high demand rates at those retailers, there are more orders of Item Type 2 waiting in load building and its waiting time gets improved when the others gets shipped out faster. There is no change in load building time for any item type at retailer 4. The largest lower weight limit may contribute to this. Change of waiting time in the backlogging queue is negligible. In Case 4 where priority is given to both backlogging and load building queue, the change of load building time gets even more complicated, since the orders with smaller amounts come out of the backlogging queue first and join the load building queue. A load may be formed and delivered before an order with small total weight but larger order quantity gets filled in the backlogging queue.

The comparison of the three case results with the base case for total lead time is summarized in Table 6. Other important statistics collected in the experiments are the aggregate fill rates within the network. These are summarized in Table 7. It appears that Case 3 with ranked backlogging and load building queues has the best performance.

Table 6: Comparison of Results

Retailer	#Item Types w/. Less Backlogging Time			#Item Types w/. Less Load Building Time			#Item Types w/. Less Total Lead Time			Avg. % Change in Total Lead Time		
	Case			Case			Case			Case		
	1	2	3	1	2	3	1	2	3	1	2	3
A	3	0	3	2	3	3	3	3	3	-3%	6%	-9%
B	4	0	4	3	3	3	4	3	4	-14%	-3%	-17%
C	3	0	3	1	2	2	2	2	3	-1%	-1%	-3%
D	3	0	3	3	2	2	3	2	3	-11%	-3%	-14%
E	3	0	3	1	3	1	3	3	3	6%	-1%	5%
F	3	0	3	1	0	1	3	3	3	-1%	0%	-1%

Table 7: Aggregate Fill Rates within Network

	Base Case	Case 1	Case2	Case 3
Whole Network	0.36	0.39	0.37	0.41
Warehouse	0.17	0.21	0.17	0.21
Item Type 1 at Warehouse	0.84	0.87	0.84	0.87
Item Type 2 at Warehouse	0.92	0.92	0.92	0.92
Item Type 3 at Warehouse	0.90	0.91	0.90	0.91
Item Type 4 at Warehouse	0.91	0.92	0.91	0.92
Retailer A	0.43	0.37	0.41	0.46
Retailer B	0.40	0.55	0.43	0.58
Retailer C	0.44	0.38	0.44	0.39
Retailer D	0.35	0.47	0.37	0.48
Retailer E	0.31	0.24	0.30	0.25
Retailer F	0.31	0.39	0.31	0.39

5 SUMMARY AND FUTURE WORK

This paper presents methods for simulating the backlog and load building queues within a multi-echelon supply chain. Building on previous work, the paper examines methods for prioritizing the backlog and load building queues to more realistically model warehouse processing of orders prior to shipping to retailers. By processing backlogged demands based on smallest order quantity, a priority is given to those demands that can more quickly be filled directly from on-hand supplies. In addition, by processing filled demands waiting for shipping to retailer locations by smallest weight (i.e. forming loads in a greedy manner) a priority ordering is given to orders waiting to be shipped. The results indicate that these two simple processing rules can have a significant effect on the waiting time components of the lead time experienced by the retailers within the multi-echelon network. The preliminary results indicate that improvements can be obtained for waiting time and fill rate under these rules; however, more study is needed to fully understand the interactions between the rules. It should be clear that since these rules effect the lead time that they should be considered when optimally setting policy parameters at the warehouse and retail levels. Future work should examine, both in a simulation and analytical setting, not only the effects of these queuing rules on the mean lead time but also the variance of the lead time. A full experimental analysis would be needed in this case. In addition, future work should include developing analytical models for the backlog and load building queues so that their effects can be incorporated into optimization models for the optimal setting of policy parameters in multi-echelon networks.

ACKNOWLEDGMENTS

This work would not have been possible without the hard work of previous students who worked on aspects of the supply chain packages of the JSL (Vijith Varghese, Mehmet Miman, Hin-Tat Chan, Yanchao Liu, Soncy Thomas, Shikha Nangia, and Yisha Xiang) and funding from the National Science Foundation, the U. S. Navy, and the U. S. Air Force. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of any particular funding agency.

REFERENCES

Axsater, S. 2006. Inventory Control, 2nd Edition, Springer Science + Business Media, LLC, New York, NY.

Çetinkaya, S., and C.-Y. Lee. 2000. Stock replenishment and shipment scheduling for vendor managed inventory systems", *Management Sciences*, vol. 20. No. 2. Pp. 217-232.

Çetinkaya, S., F. Mutlu, and C.-Y. Lee. 2006. "A comparison of outbound dispatch policies for integrated inventory and transportation decisions", European Journal of Operational Research, 171, 1094-1112.

Çetinkaya, S., E. Tekin, and C.-Y. Lee. 2008, "A stochastic model for joint inventory and outbound shipment decisions", *IIE Transactions*, vol. 40, pp. 324-340.

Chen, F. and Y. Zheng. 1994. "Evaluating echelon stock (R, nQ) policies in serial production/inventory systems with stochastic demand", *Management Science*, vol. 40, no. 10, pp. 1262-1275.

Chiang, C. and W. Chiang. 1996. "Reducing inventory costs by order splitting in the sole sourcing environment", *Journal of the Operational Research Society*, vol. 47, no. 3, pp. 446-456.

Clark, A, and H. Scarf. 1960. "Optimal policies for a multi-echelon inventory problem", *Management Science*, vol. 6, pp. 475-490

Cohen, M. A., P. V. Kamesam, P. Kleindorfer, H. Lee, and A. Tekerian. 1990. "Optimizer: IBM's multi-echelon inventory system for managing service logistics", Interfaces, 20(1), January-February, 65-82.

Gümüs, A. T. and A. F . Güner. 2007. "Multi-echelon inventory management in supply chains with uncertain demand and lead times: literature review from an operational research perspective", *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 221, no. 10, pp. 1553-1570.

Hingginson, J.K., and J.H. Bookbinder. 1995. "Markovian decision processes in shipment consolidation", *Transportation Science*, 29 (3), 242-255.

Iyer, A.V. 2001. "Inventory Cost Impact of Order Processing Priorities Based on Demand Uncertainty", *Naval Research Logistics*, vol. 49, no. 4. Pp. 376-390.

Magableh G., M. D. Rossetti, and S. Mason. 2005. "Modeling And Analysis Of A Generic Cross-Docking Facility". In *Proceedings of the 2005 Winter Simulation Conference*, eds. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Matheus, P, and L Gelders. 2000. "The (R, Q) inventory policy subject to a compound Poisson demand pattern", *International Journal of Production Economics*, vol. 60, pp. 307-317.

Mason, S.J., R.D. Meller, S.E. Root, and S. Garman. 2009 Assessment of Vendor Contract Impacts on Retail Logistics, Final Report to CELDi and Sam's Club Stores.

Muckstadt, J. A. 2005. *Analysis and Algorithms for Service Parts Supply Chains*, Springer Science+Media, Inc.

Sherbrooke, C. C. 1968. "METRIC: A multi-echelon technique for recoverable item control", *Operations Research*, 16, 122-141

Sherbrooke, C.C. 1986. "VARI-METRIC: Improved Approximations for Multi- Indenture, Multi-Echelon Availability Models", *Operations Research*, 34 (2), 311- 340.

Rossetti, M. D. 2008. "JSL: An Open-Source Object-Oriented Framework for Discrete-Event Simulation in Java", *International Journal of Simulation and Process Modeling*, vol. 4., no. 1, pp 69-87, DOI: 10.1504/IJSPM.2008.020614

Rossetti, M. D., and H. T. Chan. 2003. A Prototype Object-Oriented Supply Chain Simulation Framework. In *Proceedings of 2003 Winter Simulation Conference*, eds. S. E. Chick, P. J. Sanchez, D. M. Ferrin, D. J. Morrice. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Rossetti, M. D. and Y. Liu. 2009. "Simulating SKU Proliferation in a Health Care Supply Chain". In *Proceedings of the 2009 Winter Simulation Conference*, eds. M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Rossetti, M. D., M. Miman, V. Varghese, and Y. S. Xiang. 2006. "An Object-Oriented Framework For Simulating Multi-Echelon Inventory Systems". In *Proceedings of the 2006 Winter Simulation Conference*, eds. L. R. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Rossetti, M. D., M. Miman, and V. Varghese. 2007. "An Object-Oriented Framework for Simulating Supply Systems", *Journal of Simulation*, vol. 2, pp. 103-116.

Tempelmeir, H. 2006. *Inventory Management in Supply Networks, Problems, Models, Solutions*, Books on Demand GmbH, Norderstdedt.

Wang, X., and S. Takakuwa. 2006. "Module-based Modeling of Production-Distribution Systems Considering Shipment Consolidation" . In *Proceedings of the 2006 Winter Simulation Conference*, eds. L. R. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Silver, E. A., D. F. Pyke, and R. Peterson. 1998. *Inventory Management and Production Planning and Scheduling*, 3rd edition, John Wiley & Sons, New York.

Zheng, Y.-S., and P. H. Zipkin. 1990. "A queueing model to analyze the value of centralized inventory information", *Operations Research*, vol. 38, pp. 296-307.

Zipkin, P. H. 2000. *Foundations of Inventory Management*, McGraw-Hill Companions, Inc.

AUTHOR BIOGRAPHIES

MANUEL D. ROSSETTI is a Professor in the Industrial Engineering Department at the University of Arkansas. He received his Ph.D. in Industrial and Systems Engineering from The Ohio State University. He serves as an Associate Editor for the International Journal of Modeling and Simulation and is active in IIE, INFORMS, and ASEE. He was a WSC proceedings editor in 2004 and a co-editor for the WSC 2009 conference. He is also author of the textbook, *Simulation Modeling and Arena* published by John Wiley & Sons. His email is <rossetti@uark.edu>

YISHA XIANG is an Assistant Professor in school of business at Sun Yat-sen University, China. She received her B.S. in Industrial Engineering from Nanjing University of Aero. & Astro., China, and M.S and Ph.D. in Industrial Engineering from University of Arkansas. Her area of interest is in repairable system modeling and inventory systems. Her email is <yisha.x.zhang@gmail.com>