Proceedings of the 2003 Winter Simulation Conference
S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds.

A PROTOTYPE OBJECT-ORIENTED SUPPLY CHAIN SIMULATION FRAMEWORK

Manuel D. Rossetti
Hin-Tat Chan

4207 Bell Engineering Center
Department of Industrial Engineering
University of Arkansas
Fayetteville, AR 72701, U.S.A.

ABSTRACT

In this paper, we discuss the design, development and testing of a prototype object-oriented framework for performing supply chain simulations. We define the primary objects required for supply chain simulations and design how each of these objects are related to each other to form a supply chain network. We also present how persistence is handled for instantiating supply chain network simulations from a database. Finally, we present a small example simulation to validate and illustrate the concepts.

1 INTRODUCTION

Logistics is now a 900 billion dollar industry within the United States and results in over 3.5 trillion dollars in worldwide activity. The modeling, analysis, and optimization of logistical supply chains has become increasingly important as Internet commerce forces fundamental changes within industry. The goal of this research is to analyze and identify the fundamental elements necessary for modeling generic supply chain situations via simulation. We develop a prototype Supply Chain Simulation Framework (SCSF) to facilitate the dynamic analysis of supply chain systems. We classify and organize the modeling elements into a coherent set of objects having attributes, behaviors, and inter-relationships to form a framework for simulating supply chains. In addition, this research also provides a standardized model of the object-oriented supply chain framework using the Unified Modeling Language (UML) for documentation and dissemination of the framework. We tested and evaluated the prototype implementation on an example simulation scenario.

In this research, the UML was used during the object oriented development process to analyze and visualize the design. We implemented the framework in a prototype form using the Java computer language to provide a proof of concept for the framework. Because of the complexity of the data elements within a supply chain, we designed and developed a persistent storage framework for storing supply chain model instances within a database. Java Data Object (JDO), the PointBase database system and the Java Forte development environment was used to develop and implement the object to relational database mapping. During the simulation phase, the Java Simulation Library (JSL) is used to provide simulation support to each supply chain element. The JSL provides resources, such as generator, event calendar, scheduler, response variables, model element, etc.

In this paper, we give a brief introduction to the area of object-oriented modeling and the research literature as it applies to our effort. We then present the Supply Chain Simulation Framework (SCSF). In this, we detail the objects, their behaviors, and their interactions. We also discuss the implementation of the framework within the Java computer language. Finally, we illustrate the use of the framework on a simple multi-echelon inventory scenario.

2 BACKGROUND

Researchers and practitioners have several definitions for the supply chain. Each definition contains common key words such as logistics network, supplier, end-customer, raw material, information, goods/products, services, and facilities. From a management view, Tan (1998) defines a supply chain as encompassing material/supply management from the supply of basic raw materials to final product; it also focuses on how firms utilize their suppliers' processes, technology and capability to enhance competitive advantage. From a logistics view, Saunders (1997) defines a supply chain as an external chain that is the total chain of exchange from original source of raw material, through the various firms involved in extracting and processing raw materials, manufacturing, assembling, distributing and retailing to ultimate end customers. Ellram (1991) defines "a supply chain as a network of firms interacting to deliver product or service to end customer, linking flows from raw material supply to final delivery". Likewise, Lee

and Billington (1992) define a supply chain as networks of manufacturing and distribution sites that procure new materials, transform them into intermediate and finished products to customers. Kopczak (1997) group suppliers, logistics services providers, manufacturers, distributors and resellers as a set of entities, through which materials, products and information flow. Based on our research, a supply chain consists of two networks, which are the relationship network between a set of elements (facilities and end-customers) which have functions associated with the creation or consumption of the products/services, and a transportation or distribution network that delivers the ultimate products/services between elements. This research project concentrates on implementing the classes involved in the relationship network within a supply chain.

Ingalls and Kasales (1999) discuss the development of Compaq Computer's Supply Chain Analysis Tool (CSCAT) based on the Arena simulation-modeling environment. The tool is able to analyze the profitability of a product for a given supply chain scenario and is able to predict the customer service levels. Umeda and Jones (1988) present an architecture for integrating supply chain simulation with enterprise information systems and decision support systems through a communication data interface. The approach taken for the supply chain simulation is hierarchical and allows modeling at the operational, tactical, and strategic levels. Bagchi et al. (1998) discuss the IBM's Supply Chain Simulator. Supply chain simulation involves the simulation of both inter-facility and intra-facility operations. For example, a supply chain simulation tool may model MRP processes, planning and scheduling, capital acquisition, labor and other resources, transportation policies, stocking policies, etc. Bagchi et al. (1998) indicates that the key to the usefulness of a supply chain simulator is the ability to translate the simulation information into costs and financial reports. This is typically achieved through the use of activity base costing models. Schunk and Plott (2000) discuss the use of Supply Solver, which was developed in an effort to provide supply chain solutions using simulation as the foundation. In addition, this simulation tool is capable of simulating a supply chain design that has many different possible combinations of process options and determines the best combination of options in term of lowest overall costing and feasible flow times.

Swaminathan (1988) discusses using a multi-agent approach to solve supply chain dynamics. He indicates that simulation does provide an effective practical approach to modeling supply chain dynamics; however, the customized simulation models are specific on a particular problem and have a limited reuse; in addition, it takes a long time to build a new simulation model. Hence, he proposes a multi-agent approach to overcome these two problems. The multi-agent system is a software component-based system, which contains a number of supply chain software agents such as retailers, manufacturers, transporters, inventory policy, etc. These agents will be activated if certain events in the supply chain system occur. Thus, each agent has the self-reactor behaviors to respond to the event occurring. Because of these advantages, our programming framework will attempt to support an agent approach.

Object-Oriented Modeling (OOM) is an approach that can describe a particular system domain down to the object level. Generally, an object-oriented system will usually be composed of many objects; these objects may have relationships with other objects. An object-oriented system can be organized by classes to build a hierarchy of objects. The Unified Modeling Language (UML) has emerged as a standard for object-oriented modeling of software development and general systems modeling. The UML has a set of graphical notations and well-defined set of semantics to enable us to depict a particular domain in object-oriented model, such as a supply chain system. An object-oriented model usually has the abstractions of real world objects. These abstractions can be described or represented in a UML model by classes, attributes, behaviors, objects, associations, and states. Within the UML, each class is indicated with a rectangle divided into three areas for the class name, attributes and operations. An object attribute is a named property of a class that describes a value held by each object of the class. Each class can have some operations. An operation is the implementation of a service that can be requested from any object of the class. Operations affect the behavior of an object instance. Associations are indicated by an adorned line between classes.

In this research, we develop an object-oriented framework. A framework presents a reusable design of all or part of a system that is represented by a set of abstract classes and the way their instances interact. The instances or components of a framework can be easily connected to make a new application. Thus, the resulting application will be efficient, easy to maintain, and reliable. Developing a robust supply chain simulation framework is our objective in this research.

3 SUPPLY CHAIN SIMULATION FRAMEWORK

In this section, we present the classes within our supply chain simulation framework. In addition, we discuss how they are organized and their behavior. The current prototype framework consists of 29 classes representing various elements within a supply chain. The set of classes is given in Table 1. In order for us to show the object-oriented conceptual model of a supply chain system, we partition the supply chain model into several small conceptual sub-models. Figure 1 illustrates the Relationship Network conceptual of the supply chain system which is the emphasis of this paper.

Table 1: Class List of SCSF

Container	Parameter
ContinuousReorderQuantity	PeriodicReorderPoint
ContinuousReorderUpToLevel	PeriodicReview
ContinuousReview	Product
Demand	ProductFamily
DemandGenerator	Region
DistributionCenter	Relationship
Facility	RelationshipNetwork
Inventory	Shipment
InventoryPolicy	Shipper
Location	StorageLocation
ManufactureringCenter	TransportationCenter
Node	Variable
Order	Warehouse
OrderGenerator	

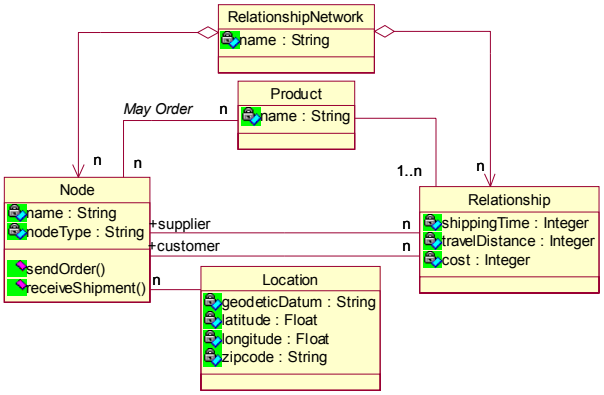

Figure 1: Conceptual Relationship Network

This conceptual network has a set of Nodes and Relationships. RelationshipNetwork is a complex system of interconnected network nodes that exchange material and information in order to provide material, products, or services to end-users. A Node in this RelationshiopNetwork can represent a Facility, Region, or OrderGenerator. These three types of Node are not in Figure 1 but they will be discussed shortly. In Figure 1, each Node may have many Relationships with another Node. A Relationship represents a conceptual connection between two Nodes and indicates the possible flow of information or material between the Nodes. Within the RelationshipNetwork, each Relationship is a unique conceptual connection because this connection indicates that a supplier supplies a customer with a specific Product. These three classes form a unique connection to identify a Relationship. In Figure 1, aggregation is used in this model to achieve the containment of Nodes and Relationships in the relationship network. From this conceptual model, a customer and a suppler may know who their suppliers and customers are. Therefore, they know to send orders to their suppliers and to send shipments to their customers. The Node class in RelationshipNetwork is attached to a Location class, which

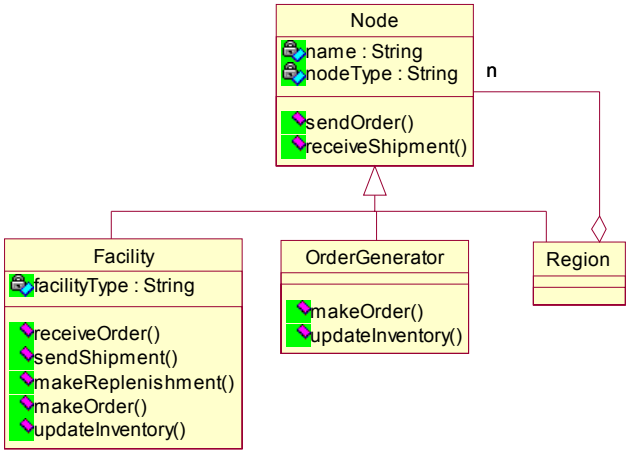

Figure 2: Types of Node

indicates the actual position of a Node in a physical network. The Physical network is not presented in this paper.

Figure 2 presents the classification of nodes. A Node represents a customer or supplier in a supply chain network. It is not a physical location in the network. Facility, OrderGenerator and Region are specializations of Node. These three types derive from the Node class so that they can inherit the Node attributes and operations. In this conceptual model, each type of Node inherits the send orders and the receive shipments method from Node. On the other hand, Facility is the only Node type that knows how to receive orders and send shipments, because it can act as a product manufacturer in a network. OrderGenerator can be considered as a single customer or as a set of aggregate customers. The conceptual model in Figure 2 shows a composite pattern representing a Region. This pattern indicates that a Region can be formed by a group of any type of these three types of Nodes. Likewise, the region is the aggregation of an entire area (zone), which can be zoned using zoning criteria such as the postal code. These aggregations provide flexibility in modeling supply chain networks.

A supply chain network consists of many facilities. Facility in the supply chain provides products or services to customers. The role of a Facility in the network is to manufacture products, distribute products, consolidate shipments, and deliver shipments. Each type of Facility knows how to receive orders from customers and send shipments to customers. Facility plays several roles in a network; as a result, we categorized a Facility into five different concepts in the framework design. They are ManufacturingCenter, DistributionCenter, TransportationCenter, Shipper, and Contractor. Each of these concepts performs different tasks in a network. The class diagram in Figure 3 shows five types of facilities.

A ManufacturingCenter is usually responsible for manufacturing finished Products or unfinished Products from raw materials. The produced Products are sent to

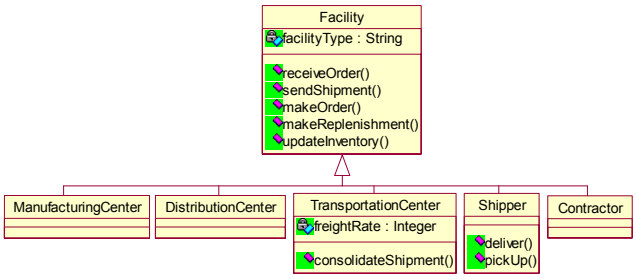

Figure 3: Types of Facility

customers whenever orders are received. The Products are made to satisfy the market demand within the network. The primary purpose of a DistributionCenter is to provide inventory replenishment and Product delivery to other facilities. DistributionCenters are primarily holding points and do not manufacture Products from raw materials. A TransportationCenter is a place where shipments of customer orders can be consolidated to obtain efficiencies in transportation. While it may hold inventory items temporarily, a TransportationCenter does not directly supply other facilities. A Shipper is mainly responsible for delivery of shipments to customers. It knows how to pick up the shipment from the TransportationCenter and deliver the shipments to customers. The contractor concept serves as a entity that has an infinite supply of material such that a contractor supplies the material after a lead time delay.

Figure 4 shows the conceptual model of Facility Inventory System, which indicates a Facility has one Warehouse Manager. A Warehouse Manager is more likely to store many Products; for each Product in a Warehouse, there will be an Inventory attached to it to keep track of the Product status in Warehouse. Therefore, Warehouse has many Inventorys; each Inventory keeps track of a Product status in every transaction.

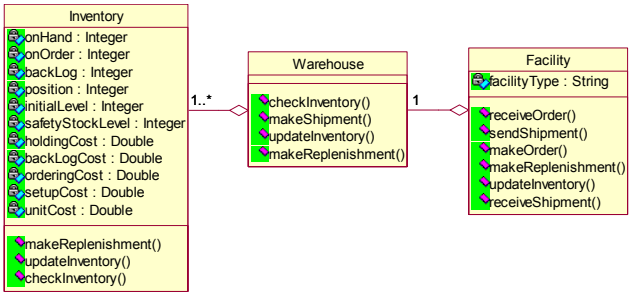

Figure 4: Facility and Inventory System

Warehouse has several important operations; checkInventory(), makeReplenishment(), and updateInventory(). These operations are to maintain the Product inventories. Whenever an order is received, the checkInventory() operation checks for every demand in the order. It sends a request to Inventory to check for the inventory status of the demanded Product. When the shipment for the replenishment order arrives, Warehouse calls updateInventory(). This operation allows the inventory to be updated and previous orders to be fulfilled if appropriate.

Each Facility in the supply chain network may store many products in its warehouse; it needs an inventory system to keep track of each product. Inventory is established and updated when Products are stored at a Facility. The Inventory tracking system helps to update inventory information during every transaction. Figure 5 is a conceptual model of Inventory and InventoryPolicy.

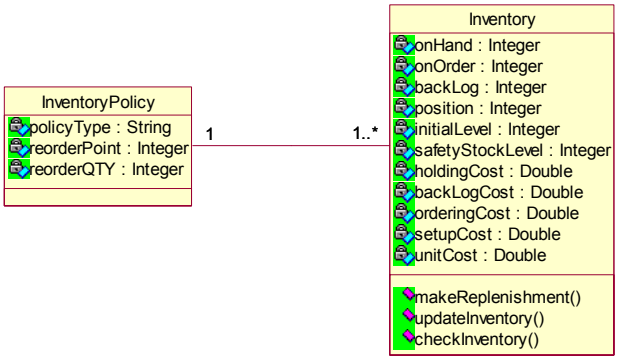

Figure 5: Inventory and Inventory Policies

Inventory policies allow the encapsulation of rules to control the associated inventory. In Warehouse, each Inventory is unique because it only keeps track of inventory status for a Product such as on-hand, backlog, position, etc. Every Inventory class is associated with one InventoryPolicy; yet this policy does not have to be unique because inventory policy only provides information about the policy type, reorder point, and reorder quantity, etc. The way we designed the InventoryPolicy class can allow the inventories in a warehouse to use the same policy for different Products. On the other hand, inventory policy is a rule, policy, or strategy that governs the re-ordering behavior for inventory of a certain type at a particular Facility. The inventory policy determines when to order and how much to order. The former is referred to the reorder point and the later is referred to the reorder quantity.

A product indicates that a specific item has been commissioned for supply from a facility on an order. Figure 6 shows the conceptual model for the relationships of a product. The relationship between Product and Node is many-to-many. This relationship indicates that each Node in the network has a set of products that it can order. On the other hand, a product can be ordered by different nodes. Figure 6 also illustrates that each facility knows to store products in its warehouse's storage locations. Each product has an inventory system attached to it to keep track of the product status in a warehouse. In the network, every Relationship can exist only when a Product is associated with it. In addition, a relationship also requires having a

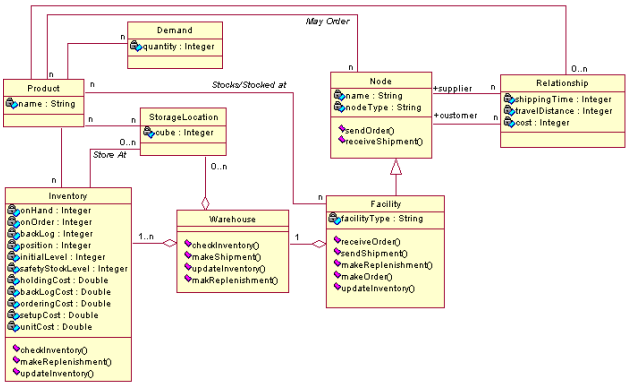

Figure 6: Relationships of Product

supplier and a customer reference. For example, a supplier supplies a product to a customer.

An order is a group of demands that have been commissioned for supply from a facility. A demand is the need for a quantity of product. A shipment is a quantity of products shipped together as part of the same cargo. Figure 7 indicates that the order has multiple relationships. A node can make many orders. Each order may have several demands in it. Once a supplier is capable of filling an order, its warehouse will make a shipment that contains the demanded products. These three supply chain elements are the entities that flow around the chain.

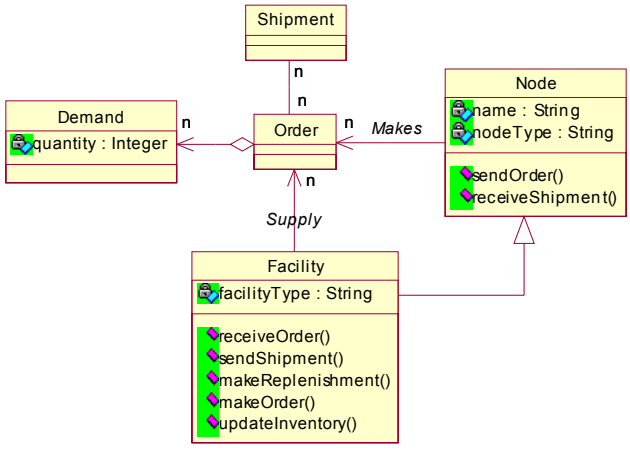

Figure 7: Relationships of Order and Demand

The conceptual model in Figure 2 shows that OrderGenerator is a type of node. The purpose of an OrderGenerator is to generate multiple demands within an order during the simulation. OrderGenerator acts as an end-customer in the relationship network. This class has several simulation attributes: time until next Order, time until last Order, time until first Order, and maximum number of Order. Each OrderGenerator has a set of DemandGenerators. Each DemandGenerator always generates demand for the same product but the demanded quantity follows a

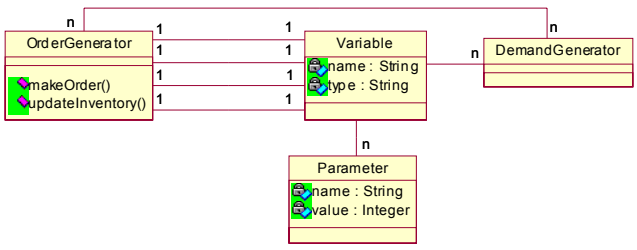

Figure 8: Order Generator Distribution

specific statistical distribution. Thus, the demanded quantity could be different in every generation. In Figure 8, each DemandGenerator has an attached Variable which provides the information about the statistical distribution. Variable specifies the distribution name and type and it stores a minimum of one parameter value as a Parameter.

4 FRAMEWORK IMPLEMENTATION AND TESTING

In the previous section, we presented an overview of the classes and their associations within the prototype supply chain network simulation framework. Supply chains are complex systems and require detailed specification for instantiating their components. We implemented the above mentioned classes within the Java computer language. Although these classes can be directly constructed into a model within the Java language, we felt that the capability to store the modeled supply chain within a database for latter re-instantiation and use would be of benefit; therefore, we provide within the framework the mechanisms to store and reconstruct supply chain network models from a persistent storage mechanism. This is similar to the work discussed by Chatfield et. al. (2001) in their explanation of SISCO (Simulator for Integrated Supply Chain Operations).

Due to space limitations, we will only give a brief discussion of how the classes in the SCSF are made persistent. We implemented a set of persistent classes that mirror their underlying non-persistent counterparts. Thus, we have two networks that work together, a persistent and a non-persistent network. The persistent network is a set of abstract classes that provide the access methods to the database. The non-persistent network is a set of abstract classes that provides interfaces to build and to simulate a supply chain network as discussed in the previous section. The classes that require persistent storage were mapped to a relational data store using a standard object to relational mapping. Persistent classes using the JDO (Java Data Object) framework are used to interact with the database. Figure 9 illustrates the overall process for instantiating a supply chain network simulation.

In order for a simulation to be executed, the supply chain network classes must have the capability to perform simulation functions (schedule events, generate random

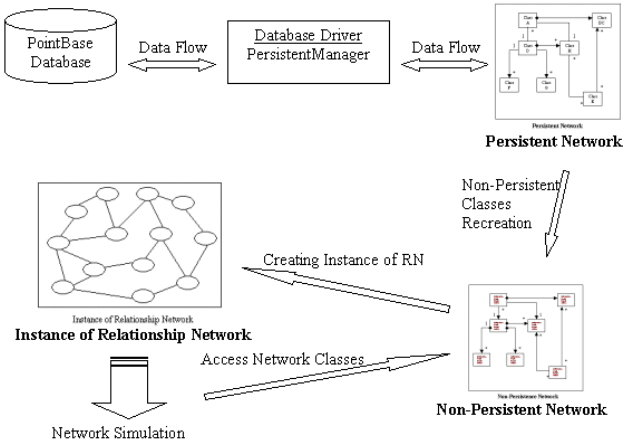

Figure 9: Flow Chart of Supply Chain Simulation Framework

numbers, etc.). The SCNF described in Section 3 could be implemented within any object-oriented language. We chose Java because of its wide acceptance, ease of use, and freely available development tools. To provide simulation capability to the SCNF, we utilized the JSL (a Java Simulation Library). Rossetti et. al. (2000) discusses the JSL. Each of the classes that need simulation capabilities are implemented in an inheritance hierarchy as illustrated in Figure 10.

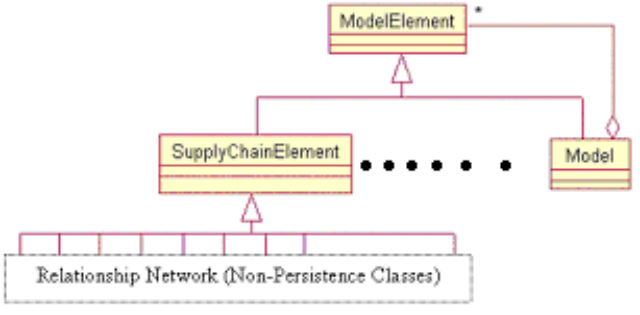

Figure 10: Inheritance Structure of Relationship Network

A ModelElement is a base class within the JSL that enables the simulation capabilities. Every simulation resource or element in JSL package is a specification of ModelElement. In the figure, the dots between SupplyChainElement class and Model class represent the rest of the JSL classes. Each class in the Relationship Network becomes a ModelElement, and each is then able to use any simulation resource. The SupplyChainElement class has an abstract method named addChildren(). The addChildren() method is responsible for ensuring that and related (child) model elements are added to the (parent) supply chain element. Each class in RelationshipNetwork will have to implement the addChildren() method. Usually, this method is used to add related ModelElements into a simulation object collection, so that all the related ModelElements will be set before every simulation. In this method, every possible child ModelElement is required to add to the collection by calling the addModelElement(object). The object is the child ModelElement. However, not every class is required to have behavior in the method, such as Product, Relationship, and InventoryPolicy. Only those classes that have a containment or creation relationship with other supply chain elements will require behavior within the addChildren() method. For example, the RelationshipNetwork creates Nodes and Relationships, and then both Node and Relationship have to be added into the simulation object collection because of the creation relationship. In this case, every object in the simulation collection will be called before every replication to reset some of the simulation attribute values.

To simulate a supply chain network model, we created a class called SCNModel, which is the trigger of the simulation. Within SCNModel, an instance of DatabaseManager is made to access the database and build a network model. In addition, the addChildren() method of RelationshipNetwork is called, and then this call will be propagated down to its children such as nodes and relationships. The addChildren() method of Node and Relationship will be called and so forth. When the addChildren() methods are done, it means all the simulation elements are added into the simulation collection, and the simulation is ready. The simulation will not run correctly unless we add all the necessary supply chain elements and simulation elements to their own simulation collection. Exhibit 1 shows the Java code of a supply chain simulation model. This code demonstrates how to get the simulation setup and run.

```
Application application = new Applica-
tion("Eddy", "SCN Test");
Model model = application.createModel();
model.turnOnDefaultReplicationReport();
model.turnOnDefaultSummaryReport();
model.turnOnDefaultTraceReport();
model.setLengthOfWarmUp(1800);
model.setLengthOfReplication(18000);
model.setNumberOfReplications(30);
SCNModel scnModel = new SCNModel();
model.addModelElement(scnModel);
model.startSimulation();
```

Exhibit 1: The Java Code of a Simulation Model

In order for the SCNModel to run, we created an Application instance. Each Application can make a Model. To collect the statistical report for a model, the Replication Report, Summary Report, and Trace Report have to be turned on. Before simulation starts, the length of warm up, run length, and number of replications for the simulation must be set. The last thing to do is to add the SCNModel as a simulation element to Model so that SCNModel will get called when the simulation starts. To run the simulation application, we just need to call the model.startSimulation() method. This method will start the simulation.

To provide initial testing for our prototype framework, we built an object-oriented simulation of a two echelon in-

ventory system and compared the results to analytically available results. The multi-echelon inventory system is illustrated in Figure 11.

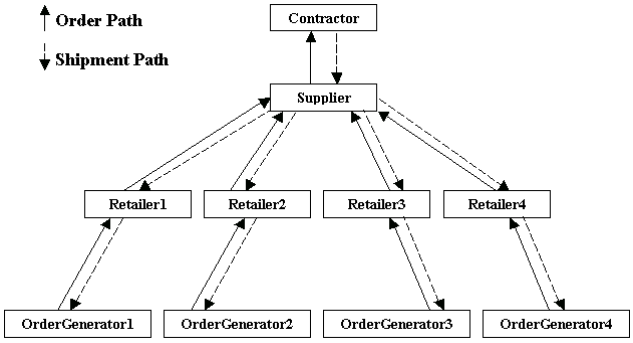

Figure 11: Multi-Echelon Inventory System

For the conditions involved within this study, we are interested in the following steady state performance measures: the expected inventory on-hand at the retailers and at the warehouse, and the expected number of items backordered at the retailers. The model is based on the work in Tee and Rossetti (2001). The demand process at the retailer is established through the specification of the time between arrivals and the demand quantity. A general renewal process with the time between orders governed by a specific probability distribution as given in the experimental design was used. Every demand from OrderGenerators was assumed to be constant (one unit of product). All unsatisfied demand will be backordered and no partial filling of an order is allowed. The replenishment lead-time of the supplier and contractor is constant (1 day), and the contractor has an unlimited capacity to supply any product.

The flowchart in Figure 12 illustrates the activities in an inventory system at a single facility location. When the inventory system receives a demand (Order from customers), the demanded quantity is determined, and then the system checks for on-hand inventory status. If the stock on-hand is sufficient to fill the demand, the quantity on-hand will be decreased by the demanded quantity. However, if the stock on- hand cannot fill the demand, the demand will be held in a queue in the inventory system. This means the order becomes a backorder. The backorders are held in a queue in Warehouse system. The backorder queue is implemented as a first-in-first-out (FIFO) queue. Whenever the on-hand, on-order, and backlog are updated, the inventory position (inventory on-hand + on-order – backorders) will be updated. The inventory position is checked each time whenever a demand is filled, or a backlog occurs, or a shipment arrives. When the inventory position falls below the reorder point, the inventory system will make a replenishment order. The inventory system can fill the backorders only when the replenishment orders (shipments) arrive. There is a time delay for the shipments to arrive.

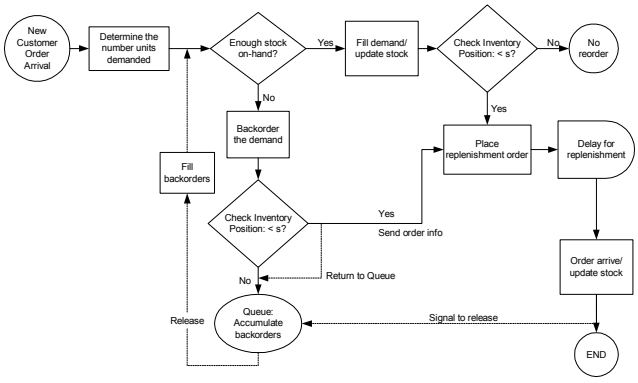

Figure 12: Flowchart of the Single Location Inventory Control Activities.

The major activities at the retailer are the same are the single location model. The only different behavior is when the retailer makes a replenishment order. The replenishment orders will be sent to its supplier (manufacturing center). If the replenishment order occurs in a supplier, the supplier will send the replenishment order to its Contractor. The Contractor is at the top of the supply chain and it has unlimited capacity on the supply of any product. This also means the Contractor does not need an inventory system. The demand process at the supplier depends on the order frequency and order quantity at the retailers. The inventory system in the supplier and retailers are similar. When the demand is filled at the supplier, a shipment will be transferred (via a delay) to retailer.

The multi-echelon exact results from Axsater (2000) and our multi-echelon results based on supply chain framework simulation model are given in Table 2 for the expected number of backorders at the retailer. The thirty-two test cases are based on multi-echelon model (4 and 32 Retailers). Comparing the results from Axsater (2000) and our simulation results in Table 2, we can conclude that there is no statistical difference between the results. Each simulation is based on 30 replications of 50 years run-length and 5 years of warm-up period. The problem descriptions and optimal policies are given in Chan (2002). Similar results were obtained for the other performance measures. Thus, the framework has be verified and validated for these types of scenarios. We conclude that our framework can model multi-echelon supply chain scenarios with little difficulty and it seems reasonable to conclude that it can be expanded to model additional supply chain scenarios.

5 SUMMARY AND CONCLUSION

In this research, we developed a supply chain simulation framework to facilitate the dynamic analysis of supply chain systems. The purpose of this research was to analyze and identify the fundamental elements necessary for modeling generic supply chain situations via simulation. We classified and organized the modeling elements into a coherent set of objects having attributes, behaviors, and inter-

Table 2: Example Two Echelon Results

Average Retailer Backorder				
	Axsater	Simulation		
Case #	Exact	Average	95% C.I.W.	Error
1	0.019	**0.018**	0.000	-0.001
2	0.009	**0.009**	0.000	0.000
3	0.005	**0.005**	0.000	0.000
4	0.034	**0.034**	0.000	0.000
5	0.118	**0.117**	0.001	-0.001
6	0.130	**0.130**	0.001	0.000
7	0.055	**0.055**	0.000	0.000
8	0.160	**0.160**	0.002	0.000
9	0.006	**0.006**	0.000	0.000
10	0.007	**0.007**	0.000	0.000
11	0.028	**0.028**	0.000	0.000
12	0.029	**0.029**	0.000	0.000
13	0.105	**0.105**	0.000	0.000
14	0.105	**0.105**	0.000	0.000
15	0.055	**0.055**	0.000	0.000
16	0.054	**0.054**	0.000	0.000
17	0.036	**0.036**	0.000	0.000
18	0.046	**0.046**	0.000	0.000
19	0.025	**0.025**	0.000	0.000
20	0.031	**0.031**	0.000	0.000
21	0.172	**0.172**	0.001	0.000
22	0.164	**0.165**	0.001	0.001
23	0.214	**0.214**	0.001	0.000
24	0.248	**0.248**	0.001	0.000
25	0.030	**0.031**	0.000	0.001
26	0.030	**0.030**	0.000	0.000
27	0.041	**0.041**	0.000	0.000
28	0.046	**0.046**	0.000	0.000
29	0.130	**0.130**	0.000	0.000
30	0.128	**0.128**	0.000	0.000
31	0.160	**0.160**	0.000	0.000
32	0.174	**0.175**	0.000	0.001

relationships to form a framework for simulating supply chains. We also provided a standardized model of the object-oriented supply chain framework using the UML for documentation and dissemination of the framework. This simulation framework can be used to model the complexity of supply chain systems. We also used our supply chain simulation framework to simulate a validated model from the literature. The results indicate that the framework can easily model the simple scenario. This simple scenario can be easily expanded to include multiple products and multi echelons. In addition, the inventory policies can vary by product, and by echelon.

Future work within the framework will involve the further development of an agent based architecture so that rules and behavior can be easily plugged into the simulation. In addition, further work is planned on making the persistent network easier to use. Finally, we are currently investigating the transportation elements within a supply chain simulations with finer detail than provided in the current prototype.

ACKNOWLEDGMENTS

This material is based upon work supported by the Arkansas Science and Technology Authority (ASTA) under Project No. 01-B-08. 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 the Arkansas Science and Technology Authority.

REFERENCES

Axsater, S. 2000. Exact Analysis of Continuous Review (R, Q) Policies in Two-Echelon Inventory Systems with Compound Poisson Demand, *Operation Research*, Vol. 48, No. 5, pp. 686-696.

Bagchi, S., S. J., Buckley, and G. Lin. 1998. Experience using the IBM supply chain simulator. In *Proceedings of the 1999 Winter Simulation Conference*, ed. Medeiros, D.J.; Watson, E.F.; Carson, J.S.; Manivannan, M.S, 1387-1394, Piscataway, New Jersey: Institute of Electrical and Electronic Engineers.

Chan, H. T. 2002. Object-Oriented Supply Chain Framework. Unpublished Masters Project Report. Department of Industrial Engineering, University of Arkansas, Fayetteville AR 72701.

Chatfield, D. C., T. P. Harrison,., and, J. C. Hayya. 2001. Sisco: A Supply Chain Simulation Tool Utilizing Silk™ And Xml. In *Proceedings of the 2001 Winter Simulation Conference*, ed. B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, 614-622, Piscataway, New Jersey: Institute of Electrical and Electronic Engineers.

Ellram, L. M. 1991. Supply chain management: The industrial organization perspective. *International Journal of Physical Distribution and Logistics Management*, Vol. 21 No. 1, 13-22.

Ingalls, R.G., and C. Kasales, 1999. CSCAT: The Compaq supply chain analysis tool. In *Proceedings of the 1999 Winter Simulation Conference*, ed. Farrington, P.A.; Black Nembhard, H.; Sturrock, D.T.; Evans, G.W, 1201-1206, Piscataway, New Jersey: Institute of Electrical and Electronic Engineers.

Kopczak, L.R. 1997. Logistics partnership and supply chain restructuring: survey results from US computer industry. *Production and Operations Management*, Vol. 6 No. 3, 226-247.

Lee, H.L, and C. Billington, 1995. The evolution of supply chain management models and practice at Hewlett-Packard. *Interfaces*, Vol. 25 No. 5, 42-63.

Rossetti, M. D., B., Aylor, R., Jacoby, A. Prorock, and A. White, 2000. SIMFONE: An object-oriented framework. In *Proceedings of 2000 Winter Simulation Conference*. ed. Joines, J.A.; Barton, R.R.; Kang, K.; Fishwick, P.A., 1855-1864. Piscataway, New Jersey: Institute of Electrical and Electronic Engineers.

Saunders, M.J. 1997. Strategic Purchasing and Supply Chain Management, Pitman.

Schunk, D. and B. Plott. 2000. Using simulation to analyze supply chain. In *Proceedings of 2000 Winter Simulation Conference*. ed. Joines, J.A.; Barton, R.R.; Kang, K.; Fishwick, P.A., 1095-1099. Piscataway, New Jersey: Institute of Electrical and Electronic Engineers.

Swaminathan, J.M. 1998. Modeling supply chain dynamics: A multi-agent approach. *Decision Sciences*, Vol. 29 No. 3, Summer, 607-632.

Tan, K.C. 2001 A framework of supply chain management literature. *European Journal of Purchasing & Supply Management*, July, 39-48.

Tee, Y.S. and M.D. Rossetti. 2001. Using Simulation To Evaluate A Continuous Review (R,Q) Two-Echelon Inventory Model. In *The Proceedings of the 6 Annual International Conference on Industrial Engineering*, November, San Francisco, CA, USA.

Umeda, S. and A. Jones. 1998. An integration test-bed system for supply chain management. In *Proceedings of the 1998 Winter Simulation Conference*, ed. Medeiros, D.J.; Watson, E.F.; Carson, J.S.; Manivannan, M.S., 1377-1385, Piscataway, New Jersey: Institute of Electrical and Electronic Engineers.

AUTHOR BIOGRAPHIES

MANUEL D. ROSSETTI, Ph. D., P. E. is an Associate 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. Dr. Rossetti has published over thirty journal and conference articles in the areas of transportation, manufacturing, health care and simulation and he has obtained over $1 million dollars in extra-mural research funding. His research interests include the design, analysis, and optimization of manufacturing, health care, and transportation systems using stochastic modeling, computer simulation, and artificial intelligence techniques. He was selected as a Lilly Teaching Fellow in 1997/98 and has been twice nominated for outstanding teaching awards. He is currently serving as Departmental ABET Coordinator. He serves as an Associate Editor for the International Journal of Modeling and Simulation and is active in IIE, INFORMS, ASEE, and SCS. Dr. Rossetti is an Associate Member of the Institute of Industrial Engineers and a member of the IIE OR Division. Dr. Rossetti is also a member of INFORMS and SCS. He will be serving as co-editor for the WSC 2004 conference. His email and web addresses are <rossetti@uark.edu> and <www.uark.edu/~rossetti>.

"EDDY" HIN-TAT CHAN graduated from Industrial Engineering at the University of Arkansas in August 2002. He received his MS degree in Industrial Engineering and bachelor degree in Computer Systems Engineering from University of Arkansas. Currently, he is working in Maynard Inc, Arkansas and can be reached via email address at <eddyhtc@yahoo.com>.