


5-2020

Simulation Modeling of Cross-Dock and Distribution Center Based Supply Chains

Ghewa Al Chall
University of Arkansas, Fayetteville

Follow this and additional works at: <https://scholarworks.uark.edu/etd>

 Part of the [Industrial Engineering Commons](#), [Industrial Organization Commons](#), [Industrial Technology Commons](#), [Operational Research Commons](#), [Operations and Supply Chain Management Commons](#), and the [Technology and Innovation Commons](#)

Citation

Al Chall, G. (2020). Simulation Modeling of Cross-Dock and Distribution Center Based Supply Chains. *Theses and Dissertations* Retrieved from <https://scholarworks.uark.edu/etd/3597>

This Thesis is brought to you for free and open access by ScholarWorks@UARK. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of ScholarWorks@UARK. For more information, please contact ccmiddle@uark.edu.

Simulation Modeling of Cross-Dock and Distribution Center Based Supply Chains

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Industrial Engineering

by

Ghewa Al Chall
Lebanese International University
Bachelor of Science in Industrial Engineering, 2017

May 2020
University of Arkansas

This thesis is approved for recommendation to the Graduate Council.

Manuel Rossetti, Ph.D.
Thesis Director

Payam Parsa, Ph.D.
Committee Member

Shengfan Zhang, Ph.D.
Committee Member

Abstract

Companies are implementing new strategies to meet the customer requirements in terms of quality, timing, and cost. One of these strategies is cross-docking, which can be defined as the process of consolidating the products coming from different suppliers, but having the same destination, with minimal handling and almost no storage between loading and unloading of the goods. The purpose of this research is to investigate the benefits of having a cross-docking facility in a supply chain.

In this research, we focus on developing discrete event simulation models using the open-source Java Simulation Library (JSL). Also, we work on augmenting an object-oriented library for simulating supply chains to include the modeling of cross-dock facilities. The modeling of a cross-dock facility includes the receiving, staging/sorting, and load building activities. Because the operational performance of the inner workings of the cross-dock is not needed, detailed modeling of the resources within the cross-dock such as the number of workers, and pieces of equipment are not included in the modeling. However, the flow, time delays, and inventory aspects are modeled because the key emphasis is on how the cross-dock affects the performance of the supply chain.

Simulation experiments are conducted to test the performance of the object-oriented library and to compare the performance of two multi-echelon inventory networks with and without cross-docking to identify the significant factors which affect the performance of the two types of supply chains.

Table of Contents

1. Introduction	1
2. Literature Review	2
3. Research Methodology	8
3.1 Conceptualization	8
3.2 Analysis.....	8
3.3 Design.....	9
3.4 Implementation and Testing	9
4. System Definition	11
4.1 Cross-Dock Based Multi-Echelon Inventory Network (CD-MEIN) Characteristics ...	11
4.1.1 Order Processing.....	12
4.1.2 Cross-dock Operations.....	13
4.2 Distribution Center Based (DC-MEIN) Multi-Echelon Inventory Network	14
4.2.1 Order Processing.....	15
4.2.2 Distribution Center Operations	15
4.3 Input Parameters	18
4.4 Key modeling issues	18
4.4.1 Outbound trucks departure.....	19
4.4.2 Transportation resources.....	19
4.4.3 Number of item types and warehouses	19

4.4.4	Demand Filling	20
5.	Modeling.....	21
5.1	Conceptual Modeling.....	21
5.2	Detailed Modeling	23
5.2.1	Modeling of a Location using LocationIfc and LocationAbstract class	25
5.2.2	Modeling of a Facility using FacilityIfc and FacilityAbstract class	27
5.2.3	Shipment and Shipment Builder Classes	32
5.2.4	Shipments Carrier Class.....	33
5.2.5	Generating Demands using DemandGenerator and GroupDemandGenerator Classes 34	
5.2.6	Network Class.....	35
6.	Performance Measures, Test Cases and Validation.....	37
6.1	Performance Measures.....	37
6.2	Testing Cases and Results.....	39
6.2.1	Total Time to Fill Demand by Each Warehouse Results.....	39
6.2.2	Aggregate Fill Rates, Item Type Fill Rates and Inventory on Hand Results.....	42
6.2.3	Total Waiting time and Inventory in Shipment Building Area Results.....	45
6.3	Validation.....	46
6.3.1	Scenario One: Increasing the Warehouses Re-order Quantity	47
6.3.2	Scenario Two: Assessing the Effect of Multiple Factors on Total Cost.....	49

7.	Conclusion and Future Work.....	54
7.1	Conclusion	54
7.2	Future Work	54
8.	References	56
9.	Appendices	58
9.1	Appendix A.....	58
9.2	Appendix B: Experimental Design Model for Scenario 2	63

List of Figures

Figure 1: Cross-Dock Based Multi-Echelon Inventory Network	12
Figure 2: Conceptual System Description for CD-MEIN.....	13
Figure 3: Distribution Center Based Multi-Echelon Inventory Network	14
Figure 4: Conceptual System Description for System 2.....	15
Figure 5: Class Diagram of LocationAbstract class	25
Figure 6: Class Diagram for FacilityAbstract Class	28
Figure 7: GroupDemandGenerator Class Methods.....	35
Figure 8: Network Class Methods	36
Figure 9: Main Effects Plot for Total Cost of DC-MEIN.....	51
Figure 10: Interaction Plots for Total Cost of DC-MEIN.....	51
Figure 11: Main Effects Plot for Total Cost of CD-MEIN.....	52
Figure 12: Interaction Plot for Total Cost of CD-MEIN	52

List of Tables

Table 1: Parameters of the Cross-Dock Multi-Echelon Inventory System (Rossetti & Xiang, 2014)	12
Table 2: Distribution Center Parameters	15
Table 3: Activities Distribution Forms	17
Table 4: List of Classes	24
Table 5: List of Interfaces	25
Table 6: Total Time (in days) to Fill Demands by Each Warehouse in Both Systems	41
Table 7: Hand Calculation for Total Time to Fill Demand by Warehouse B	41
Table 8: Fill Rate for Every Warehouse per Item Type for CD-MEIN	43
Table 9: Fill Rate for Every Warehouse per Item Type for DC-MEIN	43
Table 10: Average Inventory on hand for Every Warehouse per Item Type for CD-MEIN	44
Table 11: Average Inventory on hand for Every Warehouse per Item Type for DC-MEIN	44
Table 12: Aggregate Fill Rate for Each Warehouse in Both Systems	45
Table 13: Total Time (in days) in the Shipment Building Area per Item Type	46
Table 14: Total Time in the Shipment Building Area per Item Type	46
Table 15: Warehouses Aggregate Fill Rates after Scenario 1 for CD-MEIN	47
Table 16: Warehouses Aggregate Fill Rates after Scenario 1 for DC-MEIN	48
Table 17: Total Time to Fill Demands (in days) after Scenario 1 in CD-MEIN	49
Table 18: Total Time to Fill Demands (in days) after Scenario 1 in DC-MEIN	49
Table 19: Threshold Time, Lead Time and Time Between Demand Levels	50

List of Exhibits

Exhibit 1: Creating Item Type, External Supplier and Setting the Lead Time.....	27
Exhibit 2: Creating and Adding the Distribution-Center to the Model	30
Exhibit 3: Creating and Adding the Warehouse to the Model.....	31
Exhibit 4: Creation and Addition of Cross-Dock to the Model.....	32
Exhibit 5: Creating Shipment Carrier for External Supplier	34

1. Introduction

Companies use several product distribution networks to transport various types of goods. Most of these networks have distribution centers, which store products coming from different suppliers, and then provide the retailers with their needs. To eliminate the storing activity and reduce costs associated with it, cross-docking facilities are being utilized in many big companies like Walmart, which considers cross-docking as one of the key reasons for having high customer service (Galbreth et al., 2008).

The purpose of this thesis is to investigate the benefits of having a cross-docking facility in a supply chain. This research will focus on the development of discrete event simulation models, which will be used to compare the performance of a cross-docking facility versus the performance of a distribution center in a supply chain. The performance of the two systems will be assessed under different conditions such as demand changes. The Java simulation library (JSL) will be used to develop the simulation models and calculate the different performance metrics, which will be analyzed and evaluated through statistical analysis.

This thesis is divided into seven sections. Section two covers the literature review and background. The third section highlights the research methodology used. Then, the fourth section describes the case study, which will be modeled for each of the systems. Later, the modeling of the simulation framework is discussed in detail in section five, and its testing is presented in section six. Finally, conclusions and future work are presented in the last section.

2. Literature Review

Supply chain performance is essential for the success and the competitiveness of any company. Thus, companies are implementing new strategies to meet the customers' requirements in terms of quality, timing, and cost. One of these strategies is cross-docking, which can be defined as the process of consolidating the products coming from different suppliers, but having the same destination, with minimal handling and almost no storage between loading and unloading of the goods (Belle et al., 2012).

Different types of cross-docking can be distinguished based on several features. One of the aspects mentioned in the literature is the number of touches. In one-touch cross-docking, products are directly moved from an inbound to an outbound truck. While in multiple or two-touch cross-docking, products are unloaded and staged in the dock before being loaded in an outbound truck (Cross-docking trends report, 2008). Moreover, the distinction can also be made based on the stage where the customer is allocated to the products. In pre-distribution cross-docking, the supplier assigns the customer to the product and perform the labeling and processing activities. However, in a post-distribution cross-docking, the products are assigned to the customers and labeled at the cross-dock (Yan & Tan, 2009).

All types of cross-docking can have positive effects on the supply chain performance if implemented under the right conditions. A cross-docking facility has the ability to eliminate both storing and picking operations; thus, reducing the labor costs, inventory holding costs, and material handling costs. In addition, cross-docking decreases the delivery lead time of the products by enabling faster product flow (Galbreth et al., 2008; Ertek, 2005).

On the other hand, the implementation of cross-docking in a supply chain generates different decision problems either at the cross-dock or in the whole supply chain network. These

decision problems can be divided into three groups: strategic, tactical, or operational. For example, the design of the cross-dock and the network is a strategic problem, their planning is a tactical problem, and their scheduling is an operational problem (Buijs et al., 2014). A considerable amount of research has been done to address the different problems which exist locally at the cross-dock, and different methods were used to solve these problems like simulation and optimization.

For instance, one study focused on the scheduling problem of the different trucks and shop floor activities at a cross-dock. This study proposed a mixed integer linear programming model which was used to schedule truck arrivals and departures, and cross-dock-floor activities. The main objective of this study was to minimize the operation and transportation costs (Serrano et al., 2016). Similarly, the mixed integer linear programming method in conjunction with Lagrangean relaxation were used to solve the door assignment problem inside a cross-dock, and the main purpose was to reduce material handling costs (Nassiefa et al., 2016).

Examples on the use of modeling and simulation to solve different problems inside a cross-docking facility, include the use of discrete event simulation to determine the right size of a cross-dock, number of forklifts, number of receiving doors, and the doors layout in a certain cross-docking facility (Yang et al., 2011). Likewise, the simulation approach was used to model a generic cross-dock with all its features, and this model was used to assess the effect of demand increase on the performance of the cross-dock (Magableh & Rossetti, 2005). In addition, two simulation models were used to compare the effect of having a global staffing policy versus dedicated staffing policy on the operations of a post-distribution cross-docking facility (Cox & Rossetti, 2017). Literature reviews were done to summarize the different studies done to solve a certain cross-docking problem. More knowledge about this area can be found in (Belle et al., 2012) and (Buijs et al., 2014).

A supply chain can be defined as a network of facilities and distribution options, which operate together to make and deliver a product to the final customer (Rossetti & Xiang, 2014; Rajgopal, 2019). Cross-docking is considered as one of the most promising distribution options, which can be used to enhance the performance of the whole supply chain. As previously mentioned, existing studies have focused on solving different problems, which occur inside a cross-docking facility. However, little attention was paid to the importance of assessing the performance and the feasibility of a cross-dock as part of the whole supply chain.

One study done in this area focused on eliminating the traditional warehouse step found in a global manufacturing firm's technical consumer product supply chain; thus, improving the level of service and reducing the overall costs of the company. To achieve this end, Suh (2014) assessed the feasibility of implementing cross-docking strategy, where incoming shipments from suppliers are unloaded and sorted and directly loaded to the outbound trucks available at the dock (Suh, 2014). Suh (2014) also focused on optimizing the performance of this cross-dock by controlling different input and output parameters, using a hybrid of discrete-event and agent-based simulation model. The input parameters were: stock keeping units (SKUs) wait time, distributor order wait time, trailer full fraction and trailer wait time. While, the output parameters were: number of trailers used, SKU throughput time, less than trailer load (LTL) fill grade and percentage of LTL trailer leaving the dock. Many assumptions were made so that the simulation results can reflect the real case. For example, the demand and supply patterns were assumed to be constant. The simulation was done for 15 retailers and 20 different SKUs, with each SKU represented by a supplier. The combination of the different input parameters levels resulted in 500 different simulation runs. These simulation results helped the author in understanding the effect of the variation of each input parameter on all the output parameters. Besides, the author identified five

different cases where the cross-docking performance metric values were optimized. In one of the cases, all the output parameters were optimized, where the total number of trailers, SKU throughput time, and percentage of LTL trailer leaving the dock were minimized, and the LTL fill grade was maximized. To further interrogate the results of this case, two simulation scenarios were performed to assess the effect of different variations on the optimized performance metrics. In the first simulation, the total average wait times and trailer full fractions varied 1 or 2 days within their tolerances. Based on the simulation runs, a regression model for each performance metric was obtained, and these models highlighted that all the output parameters showed normal distributions except LTL fill grade. In the second simulation, demands from the distributors varied with $\pm 10\%$ and Any Logic's simulation was used to assess the effect of this variation on the output parameters. The results of 200 simulation runs showed that the cross-dock can perform close to the optimal setting when the demand varies within this range. The results of all the simulation runs and sensitivity analysis provided the decision-makers with the impact of each input parameter on the different output parameters. This can help in choosing the best combination of parameters needed to optimize the performance of the cross-dock, which can replace the traditional warehouse facility and enhance the performance of the whole supply chain. The methodology used in this study helps optimize and assess the performance of a cross-dock, and the Monte Carlo simulations with the sensitivity analysis performed are useful for evaluating the effect of different variables on the performance of a certain cross-docking facility. However, this study didn't show the performance measures of the distribution center facility under the same parameters and conditions. The comparison between the performance measures of the two systems is very essential for measuring the benefits of shifting from the traditional warehouse step to cross-docking.

In this paper, we discuss the development and the use of discrete event simulation models

to represent the performance of a cross-docking facility in a supply chain. The models can be used to assess the impact of demand variation and other parameters on the performance of the cross-dock. Java simulation library (JSL), which is an open source simulation library that enables discrete event simulation modeling in the Java programming language (Rossetti & Xiang, 2014), is used to model the system. More knowledge about the JSL is provided in (Rossetti, 2008).

This JSL library has packages that facilitate the modeling of multi-echelon inventory systems, which consists of two or more echelons where locations at the top serve as suppliers to locations at lower echelons. Xiang & Rossetti (2014) used these packages to highlight two key modeling issues, which can arise while modeling a multi-echelon inventory system. This study addressed the different methods, which can be used to fill backlog demands for an inventory, and this is usually referenced as backlog policy. It also assessed the different load building strategies, which can be used to fill demands waiting for transportation. The main objective of their work was to model these two processing rules suitably; thus, reducing waiting times (Xiang & Rossetti, 2014). Using these packages, similar work is done to model and assess the performance of distribution center based supply chains, and to handle the two key modeling issues mentioned in the previous study. Moreover, additional objects are developed to model a cross-dock based supply chain and evaluate its performance measures under certain conditions.

Another study completed in this area, focused on building a generic cross-docking model using ARENA software, which can be used to assess the efficiency of an individual cross-docking facility within a company's distribution network, and to examine the effect of demand increase on the performance of the cross-docking facility (Magableh & Rossetti, 2005). The performance measures used in this study were: different resources utilization, throughput times (time to process orders at cross-docking facility) and percentage of having delayed orders. This study described

different multiple modeling assumptions such as: availability of resources, inbound and outbound doors assignment, shipments characteristics, and demand variation. In this research, similar modeling assumptions and performance measures are considered. However, modeling resources like material handling equipment's and human resources, and assessing their impact on the cross-dock performance is beyond the scope of our research.

3. Research Methodology

A research methodology is a set of specific procedures or techniques, which are used to perform research. Since our main focus was to design a simulation framework for cross-dock and distribution center based supply chains, we followed a standard object-oriented modeling and analysis approach, which consists of:

- Conceptualization
- Analysis
- Design
- Implementation and Testing

3.1 Conceptualization

The first step was to conceptualize the models required to simulate the cross-dock and distribution center based supply chains. This step included creating a problem definition for the two systems, which can clearly define what is being modeled. It involved describing the objects and activities that are included in the two systems. Once the problem definition was recognized, we started thinking about the input factors and performance measures, which can help us analyze the efficiency and the differences between the two systems. The next step of this research was to analyze the problem definition to identify the potential classes needed to model the systems. This step will be described in the next section.

3.2 Analysis

In this step, we focused on identifying the major classes of the objects. Each of these objects has its attributes, behaviors, and relationships with other objects in the system. An object attribute can be described as a property or a characteristic, which should be stored and remembered. The

behaviors of the objects can be modeled using methods that either work alone or collaborate with other objects or methods to perform a specific responsibility. The collaboration between the classes can be highlighted using UML diagrams. Therefore, in this research phase, we developed basic UML diagrams to identify the major collaborations between the classes, determine the essential attributes of each object, and highlight the important methods of each class. Besides, we identified some key modeling issues that need to be addressed in the coming phases. At the end of this phase, we had enough understanding of the models and the necessary classes, which are essential to design the simulation frameworks.

3.3 Design

In this phase, we performed a deep analysis of the classes and their characteristics. The methods and their signatures were defined clearly, and the collaboration between the classes was defined and illustrated using the necessary methods. In addition, some key modeling issues were addressed in this phase to find a suitable approach for modeling them using the defined classes and relationships. After this phase, we had the conceptual basis of the simulation framework, which needs to be translated into a Java code. This was done in the implementation phase that will be described in the next section.

3.4 Implementation and Testing

The implementation phase was the most crucial in this research. We worked on mapping the system design to code, which will help us develop the simulation frameworks. The code was developed in Java, using the Java Simulation Library. After implementing the Java code, we defined test cases and validated them using text statements to track the flow of entities in the system and verify that it is working in the right way.

Throughout this research, we revisited the phases more than once, especially during the

implementation phase. For example, we had to go back to the design phase to make the necessary changes needed to handle modeling certain scenarios. The modeling of the classes and their characteristics will be described clearly in the modeling section of this document.

4. System Definition

In general, four supply chain configurations exist, and they differ by the locations that products visit before reaching the end customer. For example, in a distribution center supply chain network (DC-SCN) products are sent from the external supplier to the distribution centers and then to the customers. While in a cross-dock supply chain network (CD-SCN) products are sent from the external supplier to the cross-dock and then to the customers. In this research, the focus is on comparing the performance of these two configurations by modeling two inventory systems and assessing their performance based on common supply chain metrics.

4.1 Cross-Dock Based Multi-Echelon Inventory Network (CD-MEIN) Characteristics

As shown in Figure 1, the CD-MEIN system consists of an external supplier, a cross-docking facility, and 6 warehouses. The cross-docking facility is the location where items are received from the supplier, sorted and then sent to the different retailers. For simplicity, assume that 4 product types are stocked at the external supplier and the warehouses, and each of the products has certain weight and volume. Each of the warehouses use reorder point reorder quantity (r, Q) inventory policies, with different reorder points and reorder quantities based on item characteristics. Demand quantities vary by product type at each warehouse. The different parameters of the warehouses in this system are mentioned in Table 1, where the time between demands parameter refers to the duration between two consecutive demand requests.

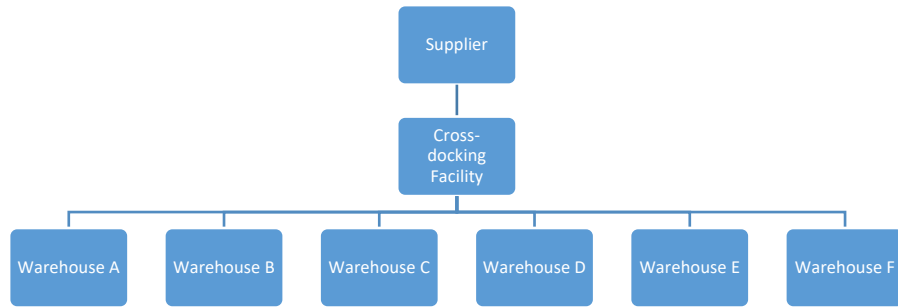


Figure 1: Cross-Dock Based Multi-Echelon Inventory Network

Table 1: Parameters of the Cross-Dock Multi-Echelon Inventory System (Rossetti & Xiang, 2014)

Warehouse	A				B				C			
Item Type	1	2	3	4	1	2	3	4	1	2	3	4
Reorder Point	200	4	3	2	50	1	2	450	500	5	10	350
Reorder Quantity	180	7	6	5	131	2	3	329	462	8	27	405
Time between Demands (days)	35.18	1.93	1.23	0.75	19.1	0.44	0.5	61.7	69.5	2.19	5.18	44.9
Warehouse	D				E				F			
Item Type	1	2	3	4	1	2	3	4	1	2	3	4
Reorder Point	500	150	15	20	250	1200	1220	650	50	300	250	300
Reorder Quantity	458	126	29	101	336	399	483	377	156	150	300	322
Time between Demands (days)	69.95	24.7	6.39	14.12	38.9	111.9	134.7	82.1	19.9	33.9	40.1	40.2

4.1.1 Order Processing

When demand occurs at any warehouse, the stock on hand is checked to determine whether the demand can be fulfilled or not. If the stock on hand is not enough, the entire order is back-ordered. The external supplier is responsible for filling the demand for a certain product and

supplies the warehouse when a backorder or replenishment order is placed. This order flow between the different locations can be shown in Figure 2.

The orders from a warehouse will not necessarily fill an entire truckload, so the shipment cannot be sent directly from supplier to warehouse due to the high transportation costs. As a result, the supplier sends orders going to multiple warehouses on the same truck. This truck goes to the cross-dock, where sorting of the products going to the same warehouse takes place. The lead time between the cross dock and the supplier is the sum of production time and transportation time, which has distribution denoted by F_1 in Table 3. All subsequent references to distributions can be found in Table 3.

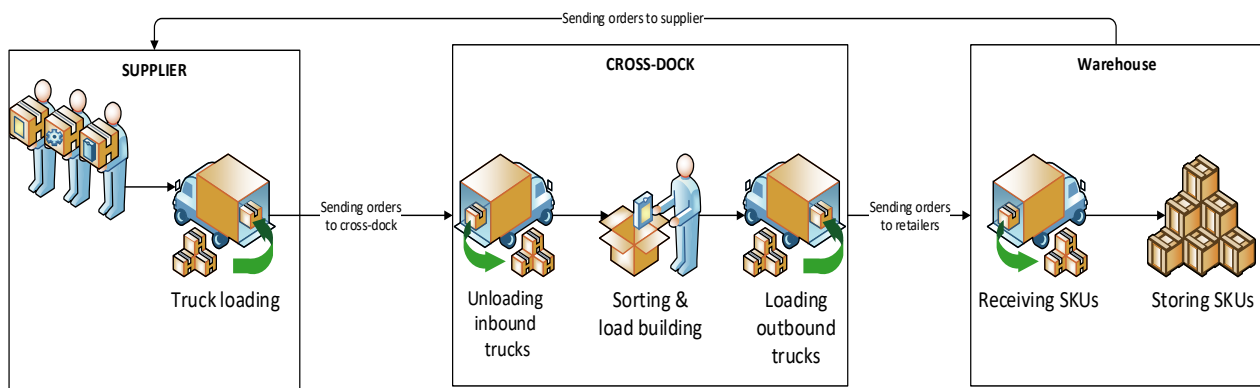


Figure 2: Conceptual System Description for CD-MEIN

4.1.2 Cross-dock Operations

When an inbound truck arrives, it is assigned to a receiving door where unloading of the products takes place. We have 5 inbound doors, and the vehicles are assigned to the doors based on FIFO rule. Once a vehicle is assigned to a door unloading starts and the unloading time has distribution F_2 . When the truck is fully unloaded, the employees start sorting the products based on their destination, and this activity has distribution F_3 . Then, loads of the sorted products are

built where each load has a maximum volume, and this loads building takes time of distribution F4. Once a full load is built, it is either moved to the staging area or directly loaded to an outbound truck going to the proper warehouse. There are 5 outbound doors available, and each one can be assigned to one truck at a time. The loading process takes the time of distribution F5. An outbound truck leaves the cross-dock when it has a full truckload, or after waiting for a maximum of 48 hours. For simplicity, assume that in all the steps, enough employees are available to perform the tasks.

4.2 Distribution Center Based (DC-MEIN) Multi-Echelon Inventory Network

As Figure 3 and Figure 4 shows, the only difference between the cross-dock network and the distribution center network is having a distribution center instead of a cross-docking facility. In this case, the warehouses place their backorders and replenishment orders to the distribution center, which uses an (r, Q) policy for each item type. The policy parameters are listed in Table 2, and the warehouses' parameters are identical to the warehouses' parameters mentioned in Table 1.

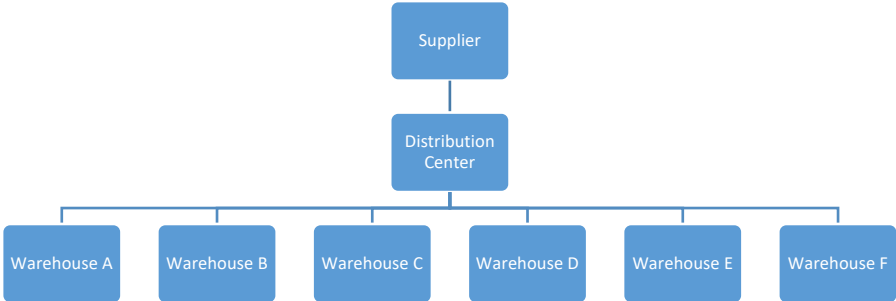


Figure 3: Distribution Center Based Multi-Echelon Inventory Network

Table 2: Distribution Center Parameters

Item Type	Reorder Point	Reorder Quantity
1	250	600
2	400	500
3	200	300
4	550	800

4.2.1 Order Processing

When an order is placed at the distribution center, and the stock on-hand is enough to fulfill this order, different activities take place. First, order picking happens where the product is picked from its storing location and moved to the dispatching area. The picking and moving activities take time, which has distribution F6. Then, dispatching takes place where products are packed and loaded to the outbound vehicle, and this takes the time of distribution F7. The transportation time between the distribution center and each of the warehouses is assumed to be of distribution F8.

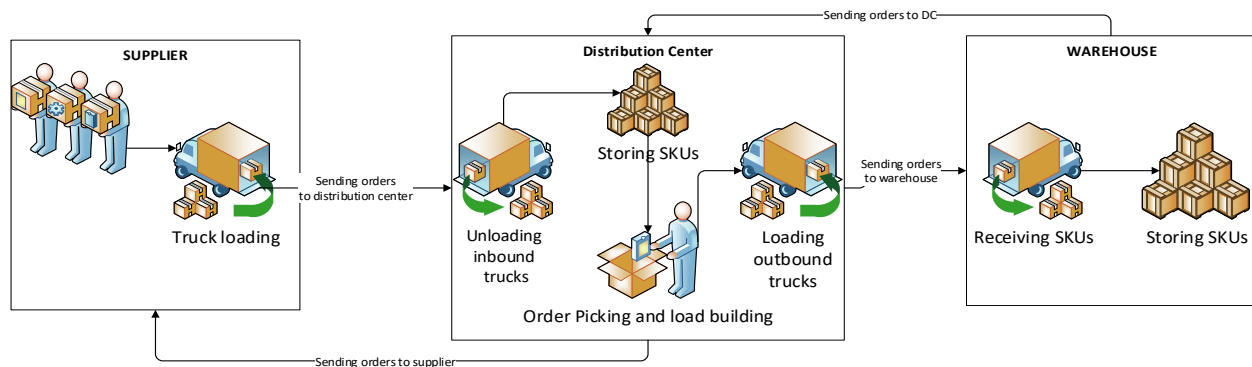


Figure 4: Conceptual System Description for System 2

4.2.2 Distribution Center Operations

Once a vehicle arrives at the facility, the vehicle details will be checked against the booking reference, and the vehicle will be allocated a location where unloading happens. Assume this

activity takes the time of distribution F9. Then, unloading the vehicle takes place at the allocated location, and this activity has distribution F10. After unloading, a checking activity may occur based on each supplier. For simplicity, assume the checking activity is not required for all the suppliers. As a result, put away activity occurs after unloading where each product is stored in a suitable location and moved through the appropriate material handling equipment. The put-away activity takes the time of distribution F11.

The probability distributions for all the activities, which occur in the two systems, are described in Table 3.

Table 3: Activities Distribution Forms

Distribution Number	Activity Description	Distribution	Source
F1	Transportation time between the cross dock and the supplier	UNIF(2,4) days	(Xiang & Rossetti, 2014)
F2	Inbound vehicles unloading time	TRIA(5,25,40) minutes	(Magableh & Rossetti, 2005)
F3	Sorting time	TRIA(10,30,60) minutes	
F4	Load building time	TRIA(15,30,60) minutes	
F5	Outbound vehicle loading time at CD	TRIA(7,26,46) minutes	(Magableh & Rossetti, 2005)
F6	Picking and moving time	TRIA(10,20,30) minutes	
F7	Outbound vehicle loading time at DC	TRIA(7,26,46) minutes	
F8	Transportation time between warehouses and DC	UNIF(2,4) hours	(Rossetti & Xiang, 2014)
F9	Vehicle checking time	TRIA(5,10,15) minutes	
F10	Unloading time for inbound vehicle at DC	TRIA(5,25,40) minutes	
F11	Put away time (per order and each order has multiple items)	TRIA(30,45,60) minutes	

4.3 Input Parameters

The simulation model developed is designed to allow the variation of several input parameters, which can affect the performance of a cross-docking facility and a distribution center.

Examples on these parameters are:

- Time between demands and demand amount: In this research, we use the word “demand” to represent the request for a certain quantity of a product placed at a warehouse. Different studies mentioned that cross-docks are suitable for products with high and stable demand (Gue, 2007). To further investigate this conclusion, different values for the time between demands, and demand amounts for each product are considered, noting that as the time between demands increases the demand level decreases because less demands are entering the system in a certain period of time.
- Number of item types: previous studies done on multi-echelon inventory systems paid little attention to the effect of the number of item types moving between the different locations on the performance of the system. This study investigates this issue to assess whether CD or DC system is better for a specific number of item types under certain conditions.
- Load building policy: processing filled demand based on different rules was also examined by Rossetti and Xiang (2010) for distribution center-based supply chain. Similarly, this research will assess the effects of these rules on both DC and CD-based supply chains to investigate the effect of the rules on the recommended configuration.

The levels for each factor will be determined later in the study after running the base case simulation.

4.4 Key modeling issues

Different key modeling issues will arise during the modeling of both systems. The most

critical ones are summarized below.

4.4.1 Outbound trucks departure

Determining when an outbound truck leaves a cross-docking facility is essential and is affected by many factors like the warehouses' rules and full truckload weight and volume limits. In this document, we assume that an outbound vehicle leaves the cross-docking facility when the lower weight or volume limit of the truckload is reached. The volume limit of the full truckload ranges between 960 ft³ and 2000 ft³ and the weight limit ranges between 5000lbs and 45000lbs. (Parsa et al., 2017). We will also assume that an outbound vehicle leaves the facility if the vehicle waiting time reaches 48 hours. Whichever case occurs first will trigger the departure of the vehicle.

4.4.2 Transportation resources

Moving products between the different locations are done using vehicles. Each location can have its transportation system, or vehicles can be shared by the different facilities. We assume that each warehouse has its vehicles responsible for transporting the products to it, and each supplier has vehicles responsible for moving products to DC or CD. We also assume that there is an unlimited number of vehicles assigned to each facility.

4.4.3 Number of item types and warehouses

A high number of warehouses, and item types can increase the complexity of the simulation. As a result, choosing the right numbers is essential for handling this issue. In this study, we will start the simulation with the numbers listed in the system description section. Then we will calibrate the model to determine the suitable numbers needed to get a significant flow of products in the systems.

4.4.4 Demand Filling

When demand occurs, and orders are placed to the external supplier or DC, in certain cases, stock on hand is not enough to fill the entire demand. This scenario can be handled in different ways. In this study, we assume that partial filling is allowed, so the available stock on hand will be used to fill part of the order, and the remaining part is back-ordered.

5. Modeling

In this chapter, we first describe the conceptual model of cross-dock and distribution center-based supply chains framework, and then discuss the framework development in detail by describing all the system elements, their roles, attributes, and relationships with each other. We will also highlight the key modeling issues faced during the development of this framework.

5.1 Conceptual Modeling

To model the cross-dock and distribution center-based supply chains, we first identified the basic elements required to build the two supply chain systems. These basic elements were determined as demand fillers, demand senders, shipment receivers, locations, facilities, demand generators, shipment builders, and shipment carriers. Each of these elements may have relationships with other elements within the framework. Most of these elements were identified either as interfaces or abstract classes in the framework, which are later used to model more concrete classes. In this section, we describe the major basic elements of the framework and highlight their major roles.

A demand filler is an object that can handle demands and fill them. While, a *demand sender* is an object that requests demands from other objects, and a *shipment receiver* is an object, which can receive shipments from other objects. In our framework, we represent a *location* as a place in the supply chain, which can fill the demands requested by other locations. A *facility* is a location that can also send demands and receive shipments; thus, a facility is a location with additional functionalities. In other words, a location is a demand filler, while a facility is a demand filler, demand sender, and shipment's receiver. For example, an external supplier is a location because it is only responsible for filling demands placed by other locations. However, a distribution center is a facility because it can send demands, fill demands, and receive shipments from other locations

or facilities. Another two major elements in the framework are shipment builders and shipment carriers. A *shipment builder* is responsible for building shipments of multiple demands going to the same destination. The shipment builder can build shipments based on many rules, which will be discussed in detail later in this chapter. Besides, a *shipment carrier* is responsible for transporting the shipments from one location to another. A *demand generator* is an object responsible for generating demands in the framework.

In any distribution-center or cross-dock based supply chain, there is a sequence of events, which occurs to fill the customer demands. For example, the sequence of events which occur in a cross-dock based supply chain can be described as:

- The demand for a specific number of items is generated using a demand generator.
- The demand arrives at a specific facility, which is the assigned filler for the demand generator.
- The facility's inventory is checked to determine whether demand can be filled directly or not. If there is enough inventory on hand, the demand will be filled immediately. If not, the demand will be back-ordered, and a replenishment order will be placed to the filler of the facility, which is, in this example, a cross-dock.
- The filler receives the replenishment order, and directly sends it to the external supplier because, in our framework, we assumed that a cross-dock does not have inventory stored in it.
- The external supplier will fill the demand and form a shipment of this demand and other demands going to the cross-dock based on the shipment building rule provided.
- The carrier will transport the shipment to the cross-dock after loading takes place.
- The cross-dock will dispatch the shipment, and consolidate the demands going to the same

destination to build a shipment and send it to the destination.

- The carrier will be loaded and then transport the shipment to the facility.
- The facility will receive the shipment, unload and dispatch it, and fill the back-ordered demands.

In the next section, we will describe the modeling of the framework objects in detail to highlight how this flow of events can be modeled.

5.2 Detailed Modeling

In this section, we illustrate the development of the framework in a more detailed way based on the conceptual model elements described in the previous section. We describe how this framework can be used to model user specific cross-dock and distribution center based supply chains. This can be easily done by implementing interfaces and abstract classes, since they allow the users to design their own supply chains by sub classing from them and overriding the methods. Throughout this section, we discuss the objects in the supply chain systems, their roles, behaviors, attributes, and relationships with other objects. We also present the implementation of specific cross-dock and distribution center based supply chains, to illustrate how this framework works and to provide a better understanding of how to use the framework.

We describe the modeling of a simple multi-echelon inventory system to illustrate the use of the framework to model this system, which consists of an external supplier, a distribution center and a warehouse. For simplicity, we assume that there is only one item type stocked at each location within this system, and both the warehouse and the distribution center, have reorder point reorder quantity (r, Q) inventory policies. The distribution center supplies the warehouse when a replenishment request is placed, and the external supplier satisfies any order placed by the distribution center. The lead time, which is the sum of the production time and transportation time

between the external supplier and the distribution center, is assumed to be 5 days. The mean time between demands arrival to the warehouse is 0.5 days. The re-order quantity for the warehouse is 3, and the reorder point is 2. While, the reorder quantity for the distribution center is 5, and its reorder point is 3. We will also describe the way to model this exact system, but with a cross-dock instead of a distribution center.

In this framework, we assume that the system is empty (at time=0), which means that no demands are flowing in the system. Besides, the initial level of inventories is set to the sum of the reorder quantity and reorder point, and all carriers are available at their facility. The simulation framework consists of thirteen classes and four interfaces, in addition to another two interfaces and four classes, which were presented in (Rossetti et al., 2008). All of these classes are shown in Table 4, and the interfaces are shown in

Table 5. This section is divided into subsections, where each subsection explains the modeling of a specific class in detail.

Table 4: List of Classes

New Classes			Previous Classes
CrossDockFacility	DistributionCenter	Shipment	InventoryHoldingPoint
ExternalSupplier	FacilityAbstract	ShipmentBuilder	Demand
LocationAbstract	GroupDemandGenerator	ShipmentsCarrier	DemandGenerator
ReceivingDock	ShippingDock	Network	ItemType
WarehouseFacility	StorageFacilityAbstract		

Table 5: List of Interfaces

New Interfaces	Previous Interfaces
FacilityIfc	DemandFillerIfc
LoactionIfc	DemandReceiverIfc
ShipmentFormingRuleIfc	DemandSenderIfc
ReceiveShipmentsIfc	

5.2.1 Modeling of a Location using LocationIfc and LocationAbstract class

In this section, we discuss the modeling of a location using LocationIfc and LocationAbstract class. We will also give an example on a LocationAbstract subclass, which is ExternalSupplier, and highlight its functionality. As mentioned before, we represented a location as a specific place in the supply chain responsible for filling demands requested by other locations. Thus, LocationIfc extends DemandFillerIfc, which was developed previously (Rossetti, Miman, & Varghese, 2008). This DemandFillerIfc allows the objects that implement it to handle demands, and be able to fill them eventually. A LocationAbstract class implements LocationIfc and extends SchedulingElement, which is a ModelElement that allows the scheduling of events.

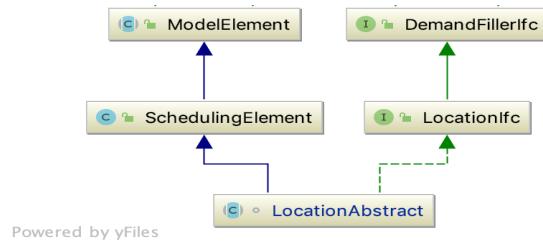


Figure 5: Class Diagram of LocationAbstract class

LocationAbstract class allows classes that extend it to have multiple characteristics. Any location can have multiple customers, which request demands from it. Each of these customers is assigned to the Location using one of the addCustomer() methods, which provide flexibility in determining the shipment building rule for this specific customer. Besides, each of these customers will have a shipment builder at the level of the Location responsible for building shipments based on the provided shipment building rule. Moreover, any location can have a carrier assigned to it responsible for transporting the shipments to the customers. In our model, we decided to collect multiple statistics for any location, which will be illustrated later in this document. Examples of these statistics are total time an item type spend inside shipment builders, and the total inventory for an item type in the shipment builders' area.

ExternalSupplier class is an example of LocationAbstract subclass, which is responsible for filling the demands requested by cross-docks and distribution centers. This class overrides the receive() and fillDemand() methods of the DemandFillerIfc to inherit its behavior. In addition to receiving and filling demands, the ExternalSupplier class has methods, which allow it to send demands to their suitable shipment builder, and then receive shipments back and direct them to the carrier to transport them. For simplicity, we assumed that the ExternalSupplier has enough inventory to fill all the demands, and have enough carriers to transport a shipment once it is built.

Error! Reference source not found. illustrates how an external supplier can be added to the model. In the hierarchy of JSL, there is a top-level model element, which is a parent for all the other model elements, so we start by creating this simulation model. Then, we create an item type using the ItemType class developed previously. Each item type needs a parent model element and a name. It can also take the volume and the weight of this item type. Later, we create an external supplier, which is an instance of the ExternalSupplier class that also needs a reference to the parent model element. We then define the distribution for the lead time of item type1 and assign it to the external supplier using the addLeadTime () method, which takes the item type and its lead time as parameters.

```
public static void Test(){
    Simulation e = new Simulation();

    Model m=e.getModel();

    ItemType myItemType1=new ItemType(m,"Type 1");
    ExternalSupplier myExternalSupplier=new ExternalSupplier(m);
    RandomVariable myLeadTime1=new RandomVariable(m, new ConstantRV(5));
    myExternalSupplier.addLeadTime(myItemType1,myLeadTime1);
}
```

Exhibit 1: Creating Item Type, External Supplier and Setting the Lead Time

5.2.2 Modeling of a Facility using FacilityIfc and FacilityAbstract class

In this subsection, we discuss the modeling of FacilityIfc, FacilityAbstract class, StorageFacilityAbstract class, and give an example on subclasses of each. As mentioned before, , we represent a facility as a location that can also send demands and receive shipments from other locations. Therefore, a FacilityIfc extends DemandSenderIfc, which allows classes that implement it to send demands to other locations, and extends the ReceiveShipmentsIfc, which allows facilities to receive shipments from other locations or facilities. As a result, FacilityAbstract class extends

LocationAbstract, since it is also a location, and implements FacilityIfc, which differentiates it from a location. The LocationAbstract class structure can be illustrated in the class diagram presented in Figure 6.

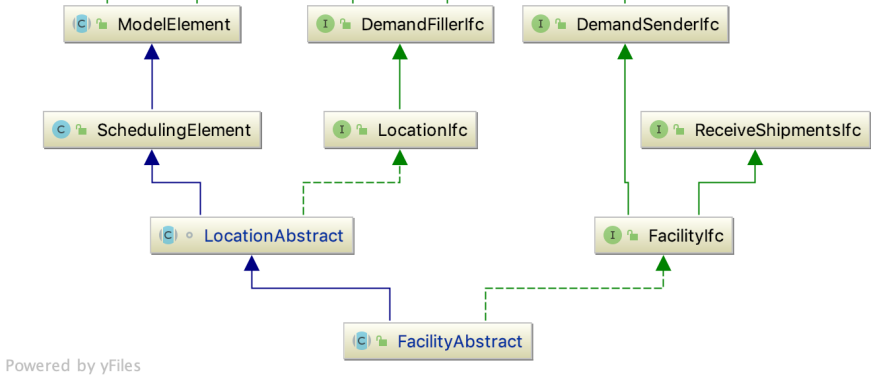


Figure 6: Class Diagram for FacilityAbstract Class

FacilityAbstract class allows each of its subclasses to have a receiving dock and a shipping dock. A receiving dock is an area where incoming shipments are unloaded and dispatched into individual demands, while a shipping dock is an area where shipments are loaded to carriers to be transported to their destination. In this framework, the connection between the facility, ShippingDock, and ReceivingDock classes is facilitated using methods, which can be overridden by subclasses to model their specific activities.

StorageFacilityAbstract class is a subclass of FacilityAbstract class, which can handle the modeling of facilities that can also have inventories for each item type. Examples on these facilities are distribution centers and warehouses. The StorageFacilityAbstract class allows a facility to have relationships with the InventoryHoldingPoint class previously developed (Rossetti, Miman, & Varghese, 2008). The InventoryHoldigPoint class holds instances of inventory for item types, and provides behavior that allows the inventories to be replenished. More description about the

InventoryHoldingPoint class can be found in (Rossetti, 2008). To facilitate the connection between the StorageFacilityAbstract class and InventoryHoldingPoint (IHP), we used inner class inside StorageFacilityAbstract class that extended DemandFillerIfc and acted as the demandFiller for the IHP. We faced a lot of challenges while trying to integrate IHP and other previously developed classes with our framework, which made us change some of our framework design. More information about this issue will be presented later in the recommendations part of this document.

DistributionCenter class is an example of StorageFacilityAbstract subclass, which has inventories for each item type stored inside the IHP. In this research, we used (r,Q) inventory policy for all inventories. When demand is requested, the DC delegates the demand to the inventory to check whether there is enough stock on hand or not. If the stock on hand is enough, the demand will be filled immediately and sent to the shipment builders' area. If not, the demand is backlogged and waits until a replenishment demand arrives to be filled. In this research, we assumed that partial filling is allowed, which means that any available stock on hand will be used to fill part of the demand, and the remaining part is back-ordered.

Exhibit 2 shows how a distribution center can be created and linked to the ES. First, we create an instance of the DistributionCenter class, then we add the item type to it, using addItemType() method. This method will communicate with other protected methods in the model to facilitate statistics collection for this specific item type. Then, we add the inventory for this item type using the addInventory() method, which takes the item type, re-order point, re-order quantity, and initial level of inventory as its parameters. Once a distribution center is added to the model, it should be linked to the external supplier that will function as its demand filler. This is done using the addCustomer() method of the external supplier, which internally creates a shipment builder for the DC, where its shipments are formed. As we mentioned before, there is more than one method

that can add customers to a specific location. These methods have different signatures since they add customers with different shipment building rules. In this exhibit, we used the default `addCustomer()` method with no shipment building rule, so we will always form shipments going to the DC regardless of the shipment weight or volume.

```
public static void Test(){
    Simulation e = new Simulation();

    Model m=e.getModel();

    ItemType myItemType1=new ItemType(m,"Type 1");
    ExternalSupplier myExternalSupplier=new ExternalSupplier(m);
    RandomVariable myLeadTime1=new RandomVariable(m, new ConstantRV(5));
    myExternalSupplier.addLeadTime(myItemType1,myLeadTime1);

    DistributionCenter myDistributionCenter=new DistributionCenter(m);
    myDistributionCenter.addItemType(myItemType1);
    myDistributionCenter.addInventory(myItemType1,3,5,8);
    myExternalSupplier.addCustomer(myDistributionCenter);
}
```

Exhibit 2: Creating and Adding the Distribution-Center to the Model

Another example of `StorageFacilityAbstract` subclass is the `WarehouseFacility` class, which represents a warehouse. The main difference between the warehouse and distribution center in this framework is the type of customers for each. A distribution center can have multiple warehouses requesting demand from it, while a warehouse will be connected to demand generators that will generate demands. Demand generators will be discussed in detail later in the `GroupDemandGenerator` subsection.

Exhibit 3 shows how a warehouse facility is made by creating an instance of `WarehouseFacility` class. Similar to the distribution center, inventory is added to the warehouse through the `addInventory()` method discussed before. The warehouse was linked to the distribution

center method using the `addCustomerWithWeightShipmentBuildingRule()` method. This notifies the distribution center should form shipments going to the warehouse based on the weight shipment building rule that has a minimum and maximum weight for shipments. This framework allows the addition of many warehouses and connecting them to the distribution center.

```
public static void Test(){
    Simulation e = new Simulation();

    Model m=e.getModel();

    ItemType myItemType1=new ItemType(m,"Type 1");
    ExternalSupplier myExternalSupplier=new ExternalSupplier(m);
    RandomVariable myLeadTime1=new RandomVariable(m, new ConstantRV(5));
    myExternalSupplier.addLeadTime(myItemType1,myLeadTime1);

    DistributionCenter myDistributionCenter=new DistributionCenter(m);
    myDistributionCenter.addItemType(myItemType1);
    myDistributionCenter.addInventory(myItemType1,3,5,8);
    myExternalSupplier.addCustomer(myDistributionCenter);

    WarehouseFacility myWarehouseA=new WarehouseFacility(m,"Warehouse A");
    myWarehouseA.addInventory(myItemType1,2,3,5);
    myDistributionCenter.addCustomerWithWeightShipmentBuildingRule(myWarehouseA,25,40);
}
```

Exhibit 3: Creating and Adding the Warehouse to the Model

An example of a subclass of `FacilityAbstract` is the `CrossDockFacility` class. In this research, we assumed that a cross-dock facility doesn't have previously-stored inventories, which can fill the demands requested by other facilities. Instead, it acts as a location where products going to the same destination are consolidated into a shipment based on specific rules. We mentioned before in the system description that warehouses send demand requests to the external supplier, which later sends the shipments back to the warehouses through the cross-dock. To make the modeling easier and make the framework more flexible, we assumed that the warehouses will send demand requests to the cross-dock, which will directly send them to the external supplier. This modeling approach allows us to easily add inventories later to the cross-dock and be able to fill demands from stock on hand inside it if necessary.

Error! Reference source not found. shows how a cross-dock facility can be created and added to the model. We start by creating an instance of CrossDockFacility class, then, we add the item type to it, using the addItemType() method. The cross-dock is linked to the external supplier using the addCustomer() method, which was used to connect the distribution center to the external supplier too. Adding warehouses and linking them to the cross-dock facility can be done similarly as presented in Exhibit 3.

```
public static void Test(){
    Simulation e = new Simulation();

    Model m=e.getModel();
    ItemType myItemType1=new ItemType(m,"Type 1");
    ExternalSupplier myExternalSupplier=new ExternalSupplier(m);
    RandomVariable myLeadTime1=new RandomVariable(m, new ConstantRV(5));
    myExternalSupplier.addLeadTime(myItemType1,myLeadTime1);

    CrossDockFacility myCrossDock=new CrossDockFacility(m);
    myCrossDock.addItemType(myItemType1);
    myExternalSupplier.addCustomer(myCrossDock);
}
```

Exhibit 4: Creation and Addition of Cross-Dock to the Model

5.2.3 Shipment and Shipment Builder Classes

A shipment is a group of demands or products consolidated together since they have the same destination. In this framework, the shipment entity is represented by the Shipment class, which extends the QObject class and holds the shipment characteristics. These characteristics include the total weight and cube of the shipment and its set of demands. Also, each shipment has a specific origin and destination, which are either facilities or locations. The Shipment class has methods, which return all of these characteristics; thus, making it easy to access them in the framework. Besides, extending the QObject class allows the shipments to be placed in queues when needed. It also permits the attachment of other objects or important attributes to any

shipment.

A shipment builder, as described before, is the object responsible for building shipments going to a specific destination based on certain shipment forming rules. In this framework, we defined three different shipment forming rules: Count, Weight, and Cube. Besides, we added the `ShipmentFormingRuleIfc`, which allows the user to define any new shipment forming rule by implementing this interface and overriding the `formShipment()` method. The Count shipment forming rule refers to forming shipments when a minimum number of items is reached. While the Weight and Cube shipment forming rules, allow shipment forming only when a minimum weight or cube is reached. The `ShipmentBuilder` class has the `receiveDemand()` method, which receives the demands from the facilities and puts them in the demand queue. Then, checking occurs to determine whether a shipment can be formed or not based on the shipment forming rule set. This class also has methods to set the parameters for the forming rules, and methods to provide the logic for shipment forming based on a specific rule.

5.2.4 Shipments Carrier Class

Every location needs a carrier to transport the shipments to their destination once they are ready. To model this carrier, we implemented the `ShipmentCarrier` class, which extends `SchedulingElement`. This class has two main methods: `transportShipment()`, and `addDestination` method. The `transportShipment()` method is used to connect the location with the shipments carrier once a shipment is built and ready to be transported. While, the `addDestination()` method is used to add the destinations, where the carrier can reach with the associated transportation time. In Exhibit 5, we can see how a shipments' carrier is created for the external supplier location by defining an instance of the `ShipmentsCarrier` class. This exhibit also shows how the `addDestination()` method was used to set the distribution center as a destination for this carrier and

assign the transportation time between the external supplier and distribution center. The same strategy can be used for creating shipment carriers and assigning destinations to them for any location.

```
public static void Test(){
    Simulation e = new Simulation();

    Model m=e.getModel();

    ItemType myItemType1=new ItemType(m,"Type 1");
    ExternalSupplier myExternalSupplier=new ExternalSupplier(m);
    RandomVariable myLeadTime1=new RandomVariable(m, new ConstantRV(5));
    myExternalSupplier.addLeadTime(myItemType1,myLeadTime1);

    DistributionCenter myDistributionCenter=new DistributionCenter(m);
    myDistributionCenter.addItemType(myItemType1);
    myDistributionCenter.addInventory(myItemType1,3,5,8);
    myExternalSupplier.addCustomer(myDistributionCenter);

    WarehouseFacility myWarehouseA=new WarehouseFacility(m,"Warehouse A");
    myWarehouseA.addInventory(myItemType1,2,3,5);

    myDistributionCenter.addCustomerWithWeightShipmentBuildingRule(myWarehouseA,25,40
);

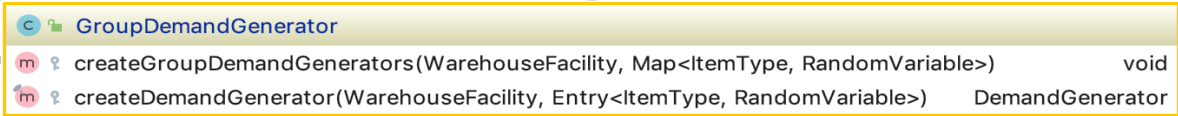
    ShipmentsCarrier myExternalSupplierCarrier=new
ShipmentsCarrier(myExternalSupplier,"ES carrier");
    RandomVariable myTransportTime=new RandomVariable(m, new UniformRV(2,4));
    myExternalSupplierCarrier.addDestination(myDistributionCenter,myTransportTime);
}
```

Exhibit 5: Creating Shipment Carrier for External Supplier

5.2.5 Generating Demands using DemandGenerator and GroupDemandGenerator Classes

A demand generator, as mentioned before, is an object that generates demands and sends them to its associated filler. Thus, a demand generator must implement the DemandSenderIfc. In this framework, we modeled the warehouses to be the fillers for the demand generators. To simplify the creation of many demand generators assigned to the same filler, we implemented the

GroupDemandGenerator class, which has only two main methods, as shown in Figure 7. The constructor of the GroupDemandGenerator class takes three parameters, which are the model element, the filler of the demand generators, and a map that has the item types with their associated time between demands. Once an instance of GroupDemandGenerator class is created, the createGroupDemandGenerators() method is called internally, which calls the createDemandGenerator() method to create a demand generator for each item type found in the given map.



```
GroupDemandGenerator
m createGroupDemandGenerators(WarehouseFacility, Map<ItemType, RandomVariable>) void
m createDemandGenerator(WarehouseFacility, Entry<ItemType, RandomVariable>) DemandGenerator
```

Figure 7: GroupDemandGenerator Class Methods

5.2.6 Network Class

To simplify the modeling of complex inventory systems, we implemented the Network class, which can build the systems more easily. This class models a network of facilities and locations based on user preferences. The top-level of this network is always an external supplier that has an infinite supply of item types and can function as a demand filler for both cross-docks and distribution centers. Figure 8 shows that the Network class contains public and protected methods, which models the relationships between this class and other objects in the framework. The first step in building the network is the addition of item types to the system using the addItemType() method. This method allows the addition of the item type with its name, weight, and volume. Once all item types are added, it's time to add the cross-dock and distribution center facilities using addCrossDock(), and addDistributionCenter() methods. As mentioned before, the top level of this network is the external supplier, which is internally created in the Network class. After adding cross-dock and distribution center facilities, they should be attached to the external supplier using the attachCrossDockToES(), and attachDistributionCenterToES() methods. The

signatures of these methods have the facility and the transportation time between the facility and the external supplier. Once we have the cross-dock and distribution center added to the network and attached to the external supplier, warehouses can be added and attached to their demand filler. This can be done using the suitable addWarehouse() method, depending on the shipment building rule. After adding any warehouse, the addInventory() method should be used to add the inventories found in this warehouse. This method should also be used to add the inventories for the distribution centers. When all the warehouses with their inventories are added to the system, group demand generators or demand generators can be attached to the warehouses using attachDemandGenartor(), and attachGroupDemandGenerator() methods. Whenever the user adds a new facility to the network, multiple checks are done internally to verify that this facility can be added to this system normally. For example, when the user uses the attachCrossDockToES() method, checks are done to determine whether the distribution center exists or not and whether it was attached before to the external supplier. These checks and others help in confirming that the right network is built.

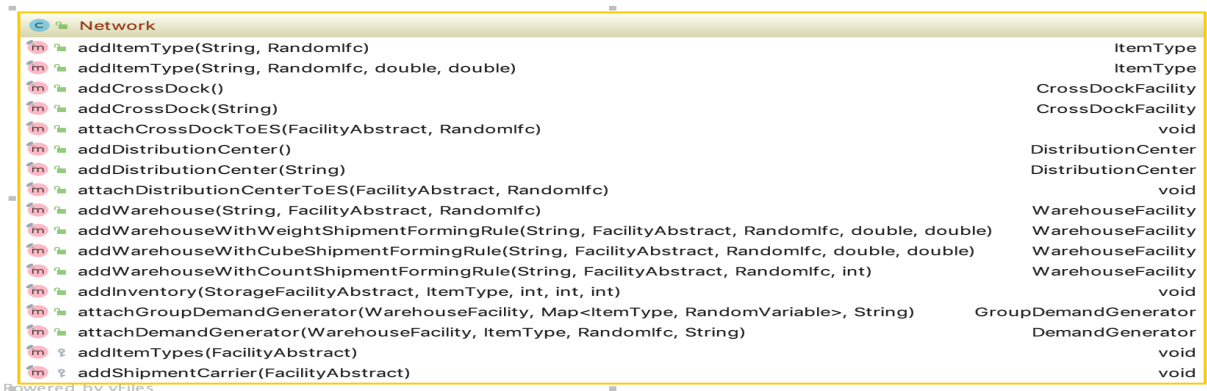


Figure 8: Network Class Methods

6. Performance Measures, Test Cases and Validation

In this section, we first describe the performance measures we use to assess the performance of the two systems. Then we focus on testing the framework and the network class to prove that all the objects are working as desired in the inventory systems. The data presented in the system definition section is used to model the cross-dock and distribution center multi-echelon inventory systems. Later, in the validation sub-section we discuss the reasons, which make these results reliable and validate that the modeling is working as intended.

6.1 Performance Measures

- Fill rate and aggregate fill rate: fill rate is the percentage of customer's demand that can be fulfilled with stock on hand without the need to backorder. This rate is considered as one of the most important customer service measures, and it can be measured for individual products in a facility and for the facility itself. In this research, we reference the fill rate of the facility as the *aggregate fill rate*, which is a demand weighted fill rate. In other words, the aggregate fill rate is the summation of the fill rates per item type multiplied by the weight of the demand of each item type. In this study, we focus on the fill rates of the warehouses in each of the systems; thus, we are able to compare the performance of the two systems and determine which system will achieve higher fill rates under specific conditions.
- Inventory on hand: is the amount of products present for sale or use in a facility (warehouse or distribution center) at a particular time. In this research, we measure the inventory on hand for every warehouse to determine its effects on other performance measures.
- Waiting time per item type in the shipment building area: is the time each item type spends in the shipment building area of the cross-dock or distribution center waiting to be shipped to its destination. This performance measure can help us determine how long the item types are

staying in the shipment building area before being shipped to their destination.

- Total inventory per item type in the shipment building area: is the total inventory found in the shipment building area of cross-dock or distribution center waiting to be consolidated into a shipment to be shipped to their destination. This performance measure allows us to determine which inventory system has a higher inventory per item type in the shipment building area.
- Total time to fill a demand in a warehouse: is the total time taken to fill a demand requested by the customer. If the warehouse has enough inventory on hand, then this time is zero. This performance measure is sensitive to the inventory level of the warehouse and the distribution center, since having higher inventory means being able to fill the demand immediately more frequently.
- Total cost: the main reason for shifting from traditional warehousing to cross-docking, is reducing the overall costs. The simulation model will determine the different costs needed to calculate the final total cost for the two systems, which is measured in \$/year. These costs are:
 - Manufacturing location (supplier) cost: it is the sum of the average inventory cost, and the cost of loading orders to either a DC or CD.
 - Total transportation cost: transportation cost is the sum of shipping cost and average in-transit inventory cost. This cost is calculated for each route; for example, there is a transportation cost for moving products from supplier to CD. Similarly, there is a transportation cost for moving the products from CD to warehouses. The sum of the transportation costs for all the routes forms the total transportation cost.
 - Cross-dock or distribution center cost: this cost is the sum of average inventory cost, cost of replenishment ordering, and cost of loading and unloading replenish at the cross-dock or distribution center.

Warehouse cost: it is the sum of average inventory cost, cost of replenishment ordering, and cost of unloading replenish at the warehouse.

6.2 Testing Cases and Results

The network class is used to build two multi-echelon inventory systems. The first one is CD-MEIN, and the second one is DC-MEIN. Each of these systems consists of an external supplier, a cross-dock or a distribution center, and six warehouses. The values of the warehouses re-order quantity, re-order point and demand rates are presented previously in Table 1. Appendix A shows how the network class is used to model the systems. The simulation time is 10 years, with 5 years warm-up period, and the number of replications is 10 for both systems. For simplicity, we assume that every two days, all products waiting at the shipment building area, are consolidated into a shipment based on their destination. The results of these simulation runs are presented in sections below, where in each section the results of a specific performance measure for both systems are presented and analyzed.

6.2.1 Total Time to Fill Demand by Each Warehouse Results

Table 6 shows the difference between the total times to fill the demand at the level of the warehouses in the two systems. The results show that the cross-dock system has higher times to fill the demands, and this is normal due to the high inventory stored in the distribution center in the second system. For example, warehouse B in the CD-MEIN has the highest time to fill the demands because it has a high demand rate, specifically for item types 2 and 3, as shown in Table 1; thus, requesting a high number of replenishment orders. However, since there is no inventory stored in the cross-dock, warehouse B is waiting for a longer time to receive replenishment orders, which is leading to a very low level of inventory on hand for item types 2 and 3 as shown in 10 and Table 11. Warehouses D, E, and F have zero time to fill demands in both systems since they

have high inventory on hand levels, as presented in 10 and Table 11 , to immediately fill the demands requested during the simulation period. To further validate the time values, a hand calculation was used to calculate the total time for warehouse B in the CD-MEIN. The total time to fill demands at warehouse B can be calculated using the following equation:

Total time to fill demand by warehouse B= lead time + waiting time at ES shipment builder + loading time + transportation time between ES and CD + unloading time + waiting time at CD shipment builder + loading time + transportation time between CD and warehouse B + unloading time.

The values of the equation components are collected from the statistics provided by the simulation, and from the input values given to the simulation like transportation, loading, and unloading times. And since these times follow certain distributions, we used the average of each distribution to calculate the total time to fill the demand by warehouse B. Table 7 summarizes the times, which when added together, give the total time to fill the demand by warehouse B in the CD-MEIN. The hand calculation gives 17.892 days as the total time to fill the demand by warehouse B, which is almost the same as the result given by the simulation that is 15.9 days. The difference between the two results is due to the variability in the times, which follow certain distributions.

Table 6: Total Time (in days) to Fill Demands by Each Warehouse in Both Systems

	CD-MEIN System	DC-MEIN System
Warehouse A	10.90061	0.563512
Warehouse B	15.918521	2.484848
Warehouse C	1.476519	0.001339
Warehouse D	0	0
Warehouse E	0	0
Warehouse F	0	0

Table 7: Hand Calculation for Total Time to Fill Demand by Warehouse B

Description	Time in days
Lead time between ES and CD	9.63
Waiting time at ES	0.99
Loading time	0.017
Transportation Time between ES and CD	3
Unloading time	0.018
Waiting time at CD	1.049
Transportation Time between CD and Warehouse B	3
Total Time to Fill demand by Warehouse B	$9.63+0.99+0.017+3+0.018+1.049+0.017+3+0.018=$ 17.892

6.2.2 Aggregate Fill Rates, Item Type Fill Rates and Inventory on Hand Results

Table 12 presents the aggregate fill rates for the warehouses in the two systems. As mentioned before, having more inventory stored in the system leads to having higher fill rates, since, the warehouse can immediately fill the demands from the inventory on hand. For example, the fill rates for warehouses D, E, and F are one for both systems because they have enough stock on hand to fill all the demands for all item types requested during the simulation, as shown in 10 and Table 11. For warehouses A, B, and C, the aggregate fill rates are higher in the DC-MEIN, and this can be justified by looking at the fill rates for each item type and inventory on hand for each warehouse in each system. For example, for warehouse A, Table 8 shows that the fill rates for item types 2, 3, and 4 are almost zero in the CD-MEIN, while Table 9 shows that the fill rates of these item types are considerably higher in the DC-MEIN. The difference in the individual fill rates is caused by the difference of stock-on hand for each item type in each system. This can be shown in Table 10 and Table 11 where the inventory on hand for item types 2, 3, and 4 are higher in the DC-MEIN; thus making the fill rates per item type, and aggregate fill rate higher. Therefore, the low aggregate fill rates for warehouse A, B, and C in the CD-MEIN are justified by having low fill rates and low inventory on hand for the item types, especially the ones with high demand levels like item type 2 and 3.

Table 8: Fill Rate for Every Warehouse per Item Type for CD-MEIN

	Item Type 1	Item Type 2	Item Type 3	Item Type 4
Warehouse A	1	0.268	0.0022	0
Warehouse B	1	0	0	1
Warehouse C	1	0.6	1	1
Warehouse D	1	1	1	1
Warehouse E	1	1	1	1
Warehouse F	1	1	1	1

Table 9: Fill Rate for Every Warehouse per Item Type for DC-MEIN

	Item Type 1	Item Type 2	Item Type 3	Item Type 4
Warehouse A	1	0.97	0.84	0.39
Warehouse B	1	0.007	0.06	1
Warehouse C	1	0.998	1	1
Warehouse D	1	1	1	1
Warehouse E	1	1	1	1
Warehouse F	1	1	1	1

Table 10: Average Inventory on hand for Every Warehouse per Item Type for CD-MEIN

	Item Type 1	Item Type 2	Item Type 3	Item Type 4
Warehouse A	294.73	0.68	0.047	0
Warehouse B	111	0	0	738.2
Warehouse C	924.91	2.45	19.41	696
Warehouse D	920.31	205	28	77
Warehouse E	504	1577	1684	988.81
Warehouse F	110	371	486	558

Table 11: Average Inventory on hand for Every Warehouse per Item Type for DC-MEIN

	Item Type 1	Item Type 2	Item Type 3	Item Type 4
Warehouse A	294.73	7.1	4.4	1
Warehouse B	110	0.007	0.13	738.2
Warehouse C	924.9	8	24	695
Warehouse D	920.3	204.9	25.96	76.69
Warehouse E	504.4	1577	1684	988.81
Warehouse F	109	371.82	486.07	558.32

Table 12: Aggregate Fill Rate for Each Warehouse in Both Systems

	CD-MEIN System	DC-MEIN System
Warehouse A	0.075293	0.654438
Warehouse B	0.015306	0.054064
Warehouse C	0.718423	0.998583
Warehouse D	1	1
Warehouse E	1	1
Warehouse F	1	1

6.2.3 Total Waiting time and Inventory in Shipment Building Area Results

Table 13 and Table 14 present the total waiting time and total inventory per item type in the shipment building area of distribution center and cross-dock. The results show that the waiting time in both areas for all item types does not exceed 2 days, since we set the threshold for the time between building shipments to be two days. Similarly, the results show that the inventory levels are almost identical in the two systems. This is also due to the time shipment building rule we assigned in this simulation.

Table 13: Total Time (in days) in the Shipment Building Area per Item Type

	Cross-Dock Shipment Builder Area	Distribution-Center Shipment Builder Area
Item Type 1	1.020071	1.135274
Item Type 2	1.011533	0.996525
Item Type 3	1.01182	1.001151
Item Type 4	1.009509	0.99581

Table 14: Total Time in the Shipment Building Area per Item Type

	Cross-Dock Shipment Builder Area	Distribution-Center Shipment Builder Area
Item Type 1	0.162437	0.177072
Item Type 2	3.362807	3.329918
Item Type 3	3.220339	3.191171
Item Type 4	1.399239	1.395677

6.3 Validation

The results presented in the previous sections validates our model in multiple ways. First, the total waiting time in the shipment building area does not exceed the threshold time, so this proves that the shipment building is working as needed. Also, the fill rates of warehouses D, E, and F are one, which is expected since their inventory on hand levels are high to immediately fill the requested demands in the simulation. Besides, the fill rates for the other warehouses are smaller in the cross-dock system, which is realistic due to having smaller inventory on hand since

replenishment orders take longer time to arrive to the warehouses as explained in the previous sections. Similarly, the total time to fill the demands is smaller in the distribution center system due to having a higher level of stock on-hand. All of these results show that the models are working as required. However, additional scenarios are tested to verify that the model is working as needed. The scenarios with their associated results are presented below.

6.3.1 Scenario One: Increasing the Warehouses Re-order Quantity

Increasing the re-order quantity for the warehouses in the CD-MEIN and DC-MEIN systems means having more inventory available at the level of the warehouses. This should increase the fill rates of the warehouses, and decrease the total time taken to fill the demands. In this scenario, we double the re-order quantity for all the warehouses and item types. Table 15 and Table 16 shows that the fill rates for the warehouses in both systems increased after doubling the re-order quantities. For example, the fill rate for warehouse A in CD-MEIN increased by 139% from 0.07 to 0.18. The reason behind the higher increase in fill rates in the DC-MEIN is again the higher level of inventory on hand in this system as described in the previous sections.

Table 15: Warehouses Aggregate Fill Rates after Scenario 1 for CD-MEIN

	Initial Fill Rate	Scenario 1 Fill Rate
Warehouse A	0.075293	0.180499
Warehouse B	0.015306	0.016
Warehouse C	0.718423	0.855982
Warehouse D	1	1
Warehouse E	1	1
Warehouse F	1	1

Table 16: Warehouses Aggregate Fill Rates after Scenario 1 for DC-MEIN

	Initial Fill Rate	Scenario 1 Fill Rate
Warehouse A	0.654438	0.804448
Warehouse B	0.054064	0.123665
Warehouse C	0.998583	1
Warehouse D	1	1
Warehouse E	1	1
Warehouse F	1	1

Error! Reference source not found. and Table 18 shows that doubling the re-order quantity of the warehouses for all item types decreased the total time to fill demands in both systems, but the time to fill demands in the CD-MEIN is still considerably high. We mentioned in previous sections that this high time is due to the low inventory on-hand and a longer time to receive replenishment orders. To further investigate this, we take warehouse B in the CD-MEIN as an example since it has the highest time to fill demands. Item types 2 and 3 have the highest demand rates in this warehouse (Time between demands are 0.44 and 0.5 days considerably), but also have very low re-order quantities. A high increase in the re-order quantities of these item types must cause a big decrease in the total time to fill demands. To prove this, we increased the re-order quantities from 2 to 40 for item type 2, and from 3 to 60 for item type 3. This led to a decrease in the total time from 15.91 to 6.62 days. This illustrates that the total times in the CD-MEIN are very high only due to the re-order quantity setting of our testing case and that they can be decreased by having more inventory on hand at the warehouse level.

Table 17: Total Time to Fill Demands (in days) after Scenario 1 in CD-MEIN

	Initial Total Time to Fill Demands	Scenario 1 Total Time to Fill Demands
Warehouse A	10.90061	8.64977
Warehouse B	15.918521	15.332512
Warehouse C	1.476519	0.721276
Warehouse D	0	0
Warehouse E	0	0
Warehouse F	0	0

Table 18: Total Time to Fill Demands (in days) after Scenario 1 in DC-MEIN

	Initial Total Time to Fill Demands	Scenario 1 Total Time to Fill Demands
Warehouse A	0.563512	0.304493
Warehouse B	2.484848	2.016644
Warehouse C	0.001339	0
Warehouse D	0	0
Warehouse E	0	0
Warehouse F	0	0

6.3.2 Scenario Two: Assessing the Effect of Multiple Factors on Total Cost

In this scenario, we examine the effect of more than one factor on the total cost of the inventory systems. These factors are the lead time of all item types, the time between demand for all item types, and threshold time, which is the minimum waiting time needed before sending a shipment to a certain location. The levels of the factors are summarized in Table 19, and the base case values for lead time and time between demand for all item types can be found in Table 1.

Table 19: Threshold Time, Lead Time and Time Between Demand Levels

	Level 1 (Low)	Level 2 (Medium)	Level 3 (High)
Threshold Time	2 days	5 days	8 days
Lead Time	Base case	+10%	+25%
Time Between demand	Base case	-20%	-40%

To perform this analysis, we designed a full factorial experimental design with a total of 27 runs. The duration of each simulation run was 10 years, and the warm-up period was 5 years. The total cost of the inventory system was calculated based on the components presented in the performance measures section. The total cost calculation is modeled in the network class; thus, allowing for the calculating of the total cost for any inventory system with any parameters. Minitab software was used to perform the experimental design and generate the plots. The full experimental model is found in Appendix B.

The effect of each factor on the total cost of DC-MEIN is presented in Figure 9. This figure shows that the threshold time and the time between demand has significant effects on the total cost of DC-MEIN. For example, as the threshold time increases the total cost decreases, which is logical, since increasing the threshold time means sending fewer shipments; thus, decreasing the shipping cost and the total cost. However, as the time between demand decreases, which means increasing the demand rate, the total cost of the DC-MEIN increases. This change in the total cost is also expected since the number of orders submitted is increasing; thus, increasing ordering costs and other costs. Figure 9 also shows that the lead time change doesn't affect the total cost, and this can be due to many reasons. One of these reasons is having a small number of examples. To further investigate the effect of these factors on the total cost of the DC-MEIN, we examined the

interaction between them. The interaction plots presented in Figure 10 show that there is no significant interaction between the three factors.

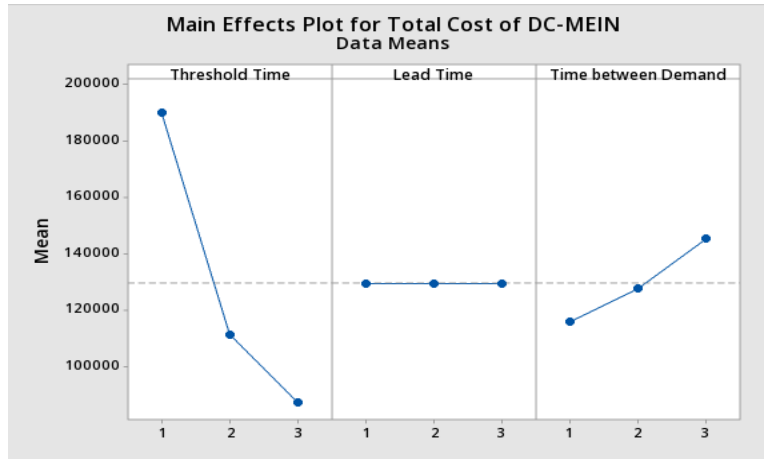


Figure 9: Main Effects Plot for Total Cost of DC-MEIN

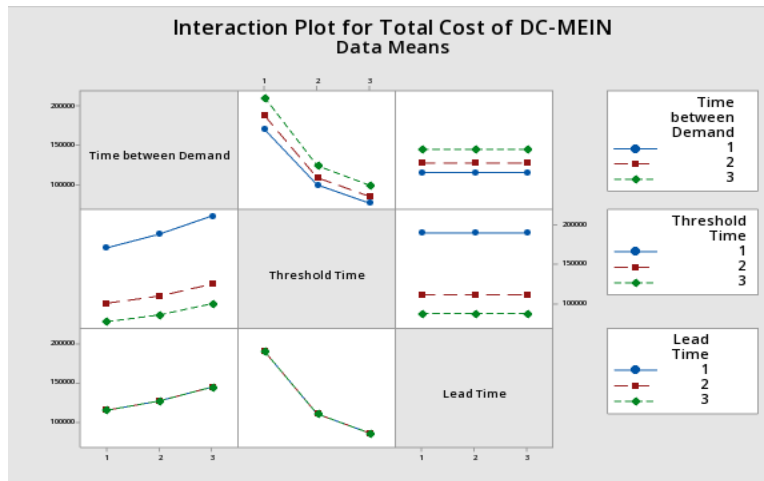


Figure 10: Interaction Plots for Total Cost of DC-MEIN

Figure 11 shows the effect of the factors on the total cost of the CD-MEIN. Similar to the DC-MEIN, the variation of the threshold time and the time between demand has significant effect on the total cost. For instance, the total cost decreased from \$220,000 to around \$75,000 when the threshold time increased from 2 to 8 days. However, the plot shows that the lead time has small

effect on the CD-MEIN when it changes from level 1 to 2. As mentioned before, the lead time effect is not highlighted in this experiment maybe due to the small number of runs. The interaction plots presented in Figure 12 show that also in the CD-MEIN there is no significant interaction between the three factors.

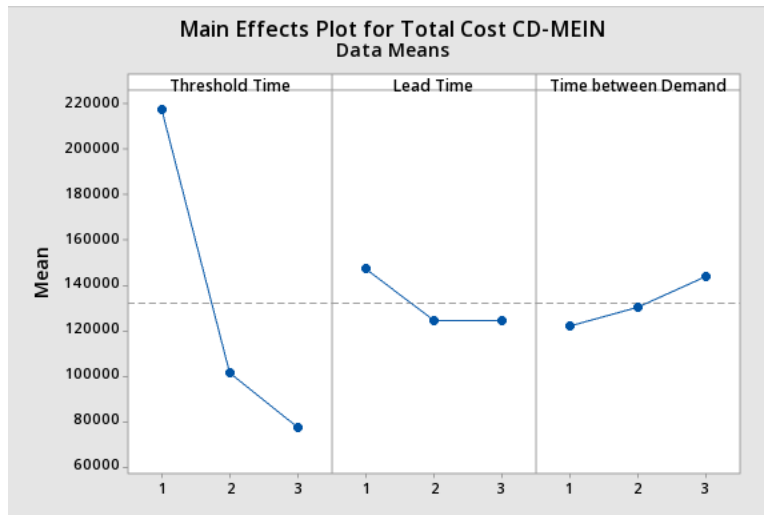


Figure 11: Main Effects Plot for Total Cost of CD-MEIN

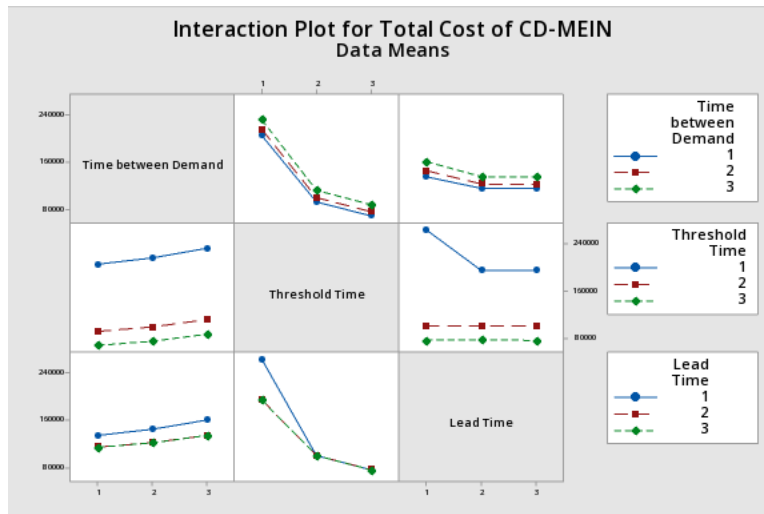


Figure 12: Interaction Plot for Total Cost of CD-MEIN

This scenario helped us validate that our model is reacting to the variations as desired and

identify that increasing the threshold time can decrease the total cost of the two systems in a large percentage.

7. Conclusion and Future Work

7.1 Conclusion

In this research, we designed and developed object-oriented simulation elements for generic cross-docks within a supply chain framework. The main purpose of this research was to analyze and identify the elements needed to model a cross-dock multi-echelon inventory network; thus, being able to assess the benefits of having a cross-dock in a supply chain. To achieve this, we organized the modeling elements into a set of objects, which have attributes, behaviors, and relationships with other objects, to form the simulation framework. We also modeled the network class, which simplify the simulation modeling of any inventory network.

To assess the performance of our framework, we used real cases of cross-dock and distribution-center inventory networks to test the elements of the framework. The performance statistics indicate that the simulation framework is working as needed and can be used to simulate real inventory networks and give reliable results.

7.2 Future Work

The future development will focus on determining under which conditions the cross-dock multi-echelon inventory network can achieve better performance measures than the distribution-center multi-echelon inventory network. This task will be done by conducting multiple experiments, which can identify the significant variables that improve the performance of the cross-dock based supply chain.

We mentioned in the modeling section that we faced multiple challenges in integrating the previous JSL objects with our framework. To prevent such challenges in future research, we will also focus on updating the previous JSL structure. One of these updates will be removing the listeners attached to the demands, which are used to determine the demand state changes and track

the flow of the demand. The use of these listeners added more challenges to the modeling of the cross-dock system because demands are not immediately filled by the cross-dock; thus, making it hard to follow the demand state changes order. Another update will be, adding more flexibility to the demand state changes order, but at the same time making sure that the right processing for the demands occurs. Besides, another important update will be, changing the way different objects are interacting together. For example, in the previous JSL design, the communication between inventory class and inventory holding point class is mainly triggered by the demand listeners. To enhance this interaction after removing the listeners, methods will be needed to improve the connection between class in the framework.

8. References

- Belle, J., Valckenaers, P., & Cattrysse, D. (2012). Cross-docking: State of the art. *Omega*, 827-846.
- Buijs, P., Vis, I., & Carlo, H. (2014). Synchronization in cross-docking networks: A research classification and framework. *European Journal of Operational Research*, 593-608.
- Cox, D., & Rossetti, M. (2017). Simulation modeling of alternative staffing and task prioritization in manual post-distribution cross docking facilities. *Winter Simulation Conference*.
- Ertek, G. (2005). A Tutorial On Crossdocking. *3rd International Logistics & Supply Chain Congress*.
- Galbreth, M., Hill, J., & Handley, S. (2008). An investigation of the value of cross docking for supply chain management. *Journal of Business logistics* , 225-240.
- Gue, K. (2007). *Warehouses Without Inventory*. Springer.
- Magableh, G., & Rossetti, M. (2005). Modeling and analysis of a generic cross-docking facility. *Winter Simulation Conference*, (pp. 1613-1620).
- Nassiefa, W., Contrerasa, I., & As'ad, R. (2016). A mixed-integer programming formulation and Lagrangean relaxation for the cross-dock doorassignment problem. *International Journal of Production Research*,, 494–508.
- Parsa, P., Rossetti, M., Zhang, S., & Pohl, E. (2017). Quantifying the benefits of continuous replenishment program for partner evaluation . *Int. J. Production Economics* , 229-245.
- Rajgopal, J. (2019). *Supply Chains: Definitions & Basic Concepts*.
- Rossetti, M. (2008). JSL: An open source object oriented framework for discrete event simulation in Java. *International Journal of Simulation and Process Modeling*, 69-87.
- Rossetti, M., & Xiang, Y. (2014). The effect of backlog and load-building processing in a multi-echelon inventory network . *ELSEVIER*, (pp. 54-66).
- Rossetti, M., Miman, M., & Varghese, V. (2008). An object-oriented framework for simulating supply systems. *Journal of Simulation*, 1-14.
- Serrano, C., Moral, J., Delorme, X., & Dolgui, A. (2016). Cross-docking Operation Scheduling: Truck Arrivals, Shop-Floor Activities and Truck departures.
- Suh, E. S. (2014). Cross-docking assessment and optimization using multi-agent co-simulation: a case study . *Springer*, 115-133.

- Xiang, Y., & Rossetti, M. (2014). The effect of backlog queue and load-building processing in a multi-echelon inventory network. *Simulation Modeling Practice and Theory*, 54-66.
- Yan, H., & Tan, S.-l. (2009). Pre-distribution and post-distribution cross-docking operations. *ELSEVIER*, 843-859.
- Yang, K. K., Balakrishnan, J., & Cheng, C. H. (2011). An analysis of factors affecting cross docking operations. *JOURNAL OF BUSINESS LOGISTICS*, 121-148.

9. Appendices

9.1 Appendix A

```
public class NetworkTestWithCD {  
    public static void buildNetworkWithDC(Model m) {  
        Network n = new Network(m, "Network");  
        Map<ItemType, RandomVariable> myMapA=new HashMap<>();  
        Map<ItemType,RandomVariable> myMapB=new HashMap<>();  
        Map<ItemType,RandomVariable> myMapC=new HashMap<>();  
        Map<ItemType,RandomVariable> myMapD=new HashMap<>();  
        Map<ItemType,RandomVariable> myMapE=new HashMap<>();  
        Map<ItemType,RandomVariable> myMapF=new HashMap<>();  
  
        //Lead Times  
        RandomVariable myLeadTime1=new RandomVariable(m, new ConstantRV(7.56));  
        RandomVariable myLeadTime2=new RandomVariable(m, new ConstantRV(10.28));  
        RandomVariable myLeadTime3=new RandomVariable(m, new ConstantRV(8.98));  
        RandomVariable myLeadTime4=new RandomVariable(m, new ConstantRV(10.41));  
  
        //Item Types  
        ItemType myItemType1=n.addItemType("Type 1",myLeadTime1);  
        ItemType myItemType2=n.addItemType("Type 2",myLeadTime2);  
        ItemType myItemType3=n.addItemType("Type 3",myLeadTime3);  
        ItemType myItemType4=n.addItemType("Type 4",myLeadTime4);  
  
        //Loading and Unloading Time in days  
        RandomVariable myUnloadingTime=new RandomVariable(m,new  
        TriangularRV(0.003,0.017,0.027));  
        RandomVariable myLoadingTime=new RandomVariable(m,new  
        TriangularRV(0.004,0.018,0.032));  
  
        //Add the cross-dock facility  
        CrossDockFacility myCrossDock=n.addCrossDock("Cross-Dock");  
        myCrossDock.setUnLoadingTime(myUnloadingTime);  
        myCrossDock.setLoadingTime(myLoadingTime);  
  
        //Attach CD to ES  
        RandomVariable myTransportTime=new RandomVariable(m, new  
        UniformRV(2,4));  
        n.attachCrossDockToES(myCrossDock,myTransportTime);  
    }  
}
```

```

//Add warehouse facility A and its inventories
WarehouseFacility myWarehouseA=n.addWarehouse("Warehouse
A",myCrossDock,myTransportTime);
n.addInventory(myWarehouseA,myItemType1,200,180,380);
n.addInventory(myWarehouseA,myItemType2,4,7,11);
n.addInventory(myWarehouseA,myItemType3,3,6,9);
n.addInventory(myWarehouseA,myItemType4,2,5,7);

//Add warehouse facility B and its inventories
WarehouseFacility myWarehouseB=n.addWarehouse("Warehouse
B",myCrossDock,myTransportTime);
n.addInventory(myWarehouseB,myItemType1,50,131,181);
n.addInventory(myWarehouseB,myItemType2,1,2,3);
n.addInventory(myWarehouseB,myItemType3,2,3,5);
n.addInventory(myWarehouseB,myItemType4,450,329,779);

//Add Warehouse facility C and its inventories
WarehouseFacility myWarehouseC=n.addWarehouse("Warehouse
C",myCrossDock,myTransportTime);
n.addInventory(myWarehouseC,myItemType1,500,462,962);
n.addInventory(myWarehouseC,myItemType2,5,8,13);
n.addInventory(myWarehouseC,myItemType3,10,27,37);
n.addInventory(myWarehouseC,myItemType4,350,405,755);

//Add warehouse facility D and its inventories
WarehouseFacility myWarehouseD=n.addWarehouse("Warehouse
D",myCrossDock,myTransportTime);
n.addInventory(myWarehouseD,myItemType1,500,458,958);
n.addInventory(myWarehouseD,myItemType2,150,126,276);
n.addInventory(myWarehouseD,myItemType3,15,29,44);
n.addInventory(myWarehouseD,myItemType4,20,101,121);

//Add warehouse facility E and its inventories
WarehouseFacility myWarehouseE=n.addWarehouse("Warehouse
E",myCrossDock,myTransportTime);
n.addInventory(myWarehouseE,myItemType1,250,336,586);
n.addInventory(myWarehouseE,myItemType2,1200,399,1599);
n.addInventory(myWarehouseE,myItemType3,1220,483,1703);
n.addInventory(myWarehouseE,myItemType4,650,377,1027);

//Add Warehouse facility F and its inventories

```

```

WarehouseFacility myWarehouseF=n.addWarehouse("Warehouse
F",myCrossDock,myTransportTime);
n.addInventory(myWarehouseF,myItemType1,50,156,206);
n.addInventory(myWarehouseF,myItemType2,300,150,450);
n.addInventory(myWarehouseF,myItemType3,250,300,550);
n.addInventory(myWarehouseF,myItemType4,300,322,622);

//Warehouse A demand times
RandomVariable myTimeA1=new RandomVariable(m,new ExponentialRV(35.18));
RandomVariable myTimeA2=new RandomVariable(m,new ExponentialRV(1.93));
RandomVariable myTimeA3=new RandomVariable(m,new ExponentialRV(1.23));
RandomVariable myTimeA4=new RandomVariable(m,new ExponentialRV(0.75));
myMapA.put(myItemType1,myTimeA1);
myMapA.put(myItemType2,myTimeA2);
myMapA.put(myItemType3,myTimeA3);
myMapA.put(myItemType4,myTimeA4);

//Warehouse B demand times
RandomVariable myTimeB1=new RandomVariable(m,new ExponentialRV(19.15));
RandomVariable myTimeB2=new RandomVariable(m,new ExponentialRV(0.44));
RandomVariable myTimeB3=new RandomVariable(m,new ExponentialRV(0.5));
RandomVariable myTimeB4=new RandomVariable(m,new ExponentialRV(61.68));
myMapB.put(myItemType1,myTimeB1);
myMapB.put(myItemType2,myTimeB2);
myMapB.put(myItemType3,myTimeB3);
myMapB.put(myItemType4,myTimeB4);

//Warehouse C demand times
RandomVariable myTimeC1=new RandomVariable(m,new ExponentialRV(69.51));
RandomVariable myTimeC2=new RandomVariable(m,new ExponentialRV(2.19));
RandomVariable myTimeC3=new RandomVariable(m,new ExponentialRV(5.18));
RandomVariable myTimeC4=new RandomVariable(m,new ExponentialRV(44.88));
myMapC.put(myItemType1,myTimeC1);
myMapC.put(myItemType2,myTimeC2);
myMapC.put(myItemType3,myTimeC3);
myMapC.put(myItemType4,myTimeC4);

//Warehouse D demand times
RandomVariable myTimeD1=new RandomVariable(m,new ExponentialRV(69.95));
RandomVariable myTimeD2=new RandomVariable(m,new ExponentialRV(24.68));
RandomVariable myTimeD3=new RandomVariable(m,new ExponentialRV(6.39));
RandomVariable myTimeD4=new RandomVariable(m,new ExponentialRV(14.12));
myMapD.put(myItemType1,myTimeD1);
myMapD.put(myItemType2,myTimeD2);
myMapD.put(myItemType3,myTimeD3);

```

```

myMapD.put(myItemType4,myTimeD4);

//Warehouse E demand times
RandomVariable myTimeE1=new RandomVariable(m,new ExponentialRV(38.95));
RandomVariable myTimeE2=new RandomVariable(m,new ExponentialRV(111.92));
RandomVariable myTimeE3=new RandomVariable(m,new ExponentialRV(134.71));
RandomVariable myTimeE4=new RandomVariable(m,new ExponentialRV(82.15));
myMapE.put(myItemType1,myTimeE1);
myMapE.put(myItemType2,myTimeE2);
myMapE.put(myItemType3,myTimeE3);
myMapE.put(myItemType4,myTimeE4);

//Warehouse F demand times
RandomVariable myTimeF1=new RandomVariable(m,new ExponentialRV(19.94));
RandomVariable myTimeF2=new RandomVariable(m,new ExponentialRV(33.98));
RandomVariable myTimeF3=new RandomVariable(m,new ExponentialRV(40.14));
RandomVariable myTimeF4=new RandomVariable(m,new ExponentialRV(40.18));
myMapF.put(myItemType1,myTimeF1);
myMapF.put(myItemType2,myTimeF2);
myMapF.put(myItemType3,myTimeF3);
myMapF.put(myItemType4,myTimeF4);

myWarehouseA.setUnLoadingTime(myUnloadingTime);
myWarehouseB.setUnLoadingTime(myUnloadingTime);
myWarehouseC.setUnLoadingTime(myUnloadingTime);
myWarehouseD.setUnLoadingTime(myUnloadingTime);
myWarehouseE.setUnLoadingTime(myUnloadingTime);
myWarehouseF.setUnLoadingTime(myUnloadingTime);

//Attach the demand generators
n.attachGroupDemandGenerator(myWarehouseA,myMapA,"Group generators
A");
n.attachGroupDemandGenerator(myWarehouseB,myMapB,"Group generators B");
n.attachGroupDemandGenerator(myWarehouseC,myMapC,"Group generators C");
n.attachGroupDemandGenerator(myWarehouseD,myMapD,"Group generators
D");
n.attachGroupDemandGenerator(myWarehouseE,myMapE,"Group generators E");
n.attachGroupDemandGenerator(myWarehouseF,myMapF,"Group generators F");
}
public static void testExperiment() {
// create the experiment to run the model
Simulation e = new Simulation();
SimulationReporter r = e.makeSimulationReporter();
buildNetworkWithDC(e.getModel());

```

```
// set the parameters of the experiment  
e.setNumberOfReplications(1);  
e.setLengthOfReplication(3650);  
e.setLengthOfWarmUp(1825);  
  
e.run();  
r.printAcrossReplicationSummaryStatistics();  
  
}
```

9.2 Appendix B: Experimental Design Model for Scenario 2

StdOrder	RunOrder	PtType	Blocks	Threshold Time	Lead Time	Time between Demand	Total Cost of DC-MEIN	Total Cost of CD-MEIN
9	1	1	1	1	3	3	211294.37	206142.88
21	2	1	1	3	1	3	99866.30	87614.04
4	3	1	1	1	2	1	170557.15	185523.87
24	4	1	1	3	2	3	99781.31	88208.93
3	5	1	1	1	1	3	211379.69	283838.05
23	6	1	1	3	2	2	85584.37	76103.27
5	7	1	1	1	2	2	188133.60	193357.74
6	8	1	1	1	2	3	211294.37	205973.60
26	9	1	1	3	3	2	85499.37	75678.28
11	10	1	1	2	1	2	109467.96	100156.78
1	11	1	1	1	1	1	170557.15	244718.25
20	12	1	1	3	1	2	85669.39	76103.53
12	13	1	1	2	1	3	124683.19	112307.05
7	14	1	1	1	3	1	170557.14	185563.73
14	15	1	1	2	2	2	109467.96	100071.69
18	16	1	1	2	3	3	124853.17	112431.75
8	17	1	1	1	3	2	188133.60	193017.37
25	18	1	1	3	3	1	76977.88	68949.33
2	19	1	1	1	1	2	188049.76	261102.48

10	20	1	1	2	1	1	100436.41	92363.30
22	21	1	1	3	2	1	76892.88	69204.49
15	22	1	1	2	2	3	124768.18	112476.93
17	23	1	1	2	3	2	109467.95	99776.60
13	24	1	1	2	2	1	100436.41	92578.26
27	25	1	1	3	3	3	99781.31	88123.75
16	26	1	1	2	3	1	100436.42	92708.14
19	27	1	1	3	1	1	76977.88	69204.52