UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

DYNAMIC OPTIMIZATION OF SERVICE PART INVENTORY CONTROL POLICY THROUGH APPLIED DATA MINING AND SIMULATION

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

degree of

Doctor of Philosophy

By

EUGENE A. BEARDSLEE
Norman, Oklahoma
2007

UMI Number: 3256647

UMI Microform 3256647

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

DYNAMIC OPTIMIZATION OF SERVICE PART INVENTORY CONTROL POLICY THROUGH APPLIED DATA MINING AND SIMULATION

A DISSERTATION APPROVED FOR THE
SCHOOL OF INDUSTRIAL ENGINEERING

BY

_____
Theodore B. Trafalis, Ph.D.

_____
F. Hank Grant, Ph.D.

_____
S. Lakshmivarahan, Ph.D.

_____
Scott A. Moses, Ph.D.

_____
Suleyman Karabuk, Ph.D.

Acknowledgements

I would like to thank all the individuals who encouraged me and inspired me to embark on the educational journey that has brought me to this point. I hold deep appreciation and love for my parents Charles and Muriel Beardslee for the years of sacrifice, service and guidance they have showered upon me. I greatly appreciate the time, guidance and friendship of my advisor, Dr. Theodore Trafalis and the time and attention my committee members have devoted to my education and advancement. I extend a special thanks to Dr. Hank Grant who introduced me to the Industrial Engineering program at OU, and for countless hours of mentoring.

And to Lisa, my patient, loving and faithful wife, without whom I would not have been able to complete this effort, I offer my sincere heartfelt thanks, and continuing love.

Table of Contents

List of Tables

List of Figures

Abstract

This research defines a novel approach for associating inventory item behavior, focusing initially on demand patterns, with an optimal inventory control policy. This method relies upon the definition of typical service part inventory demand patterns and the ability of data mining algorithms to classify inventory transaction data into one of these defined demand patterns. To facilitate this data mining effort, a simulation which creates archetypal inventory demand time series is proposed as the training data source for the data mining task. Actual service part inventory transactions thus classified will be used in a separate service part inventory simulation, modeling a multi-item inventory controlled using a set of common stochastic inventory control policies. Through simulation optimization, using simultaneous perturbation stochastic approximation (SPSA), an optimal demand-pattern to control-policy pairing is sought. The resulting set of optimal pairings will then be used to determine the optimal policy which should be applied to actual service part inventory items after performing demand classification data mining of the actual inventory transaction time series. Improving the efficiency of inventory management within the maintenance and repair service business area holds great promise for reducing inventory investment and improving customer service. Ideally, application of this research could enable an inventory management system which supports the use of multiple concurrent and dynamic inventory management policies focused on reducing inventory cost and increasing customer service and complex equipment availability.

Chapter 1

Introduction and Overview

Research Objective

The research presented here seeks to establish both a method and a set of guidelines for optimizing the cost and performance of a service part inventory. The method under analysis is a cooperative use of both simulation and data mining focused on discovering an optimized pairing of observed demand data streams and applied inventory policy. Through careful application of this method, a set of optimized demand-structure to control-policy pairings can act as guidelines which the inventory manager can dynamically apply to a large number of service part inventory items with the help of a properly configured information system.

While the goal of optimizing inventory management policy is not a new idea, five concepts introduced by this research are novel and hold great promise for improving service part inventory management. First, using simulation of repair service demand processes, demand transaction time series are generated to represent archetypal demand structures. These time series are then used as training data for input to data mining algorithms. Once the data mining models are trained and tested, they are used to classify unseen repair service inventory transactions into categories represented by each of the archetypal demand structures. Second, the applied results of this research will provide a method of pairing classified demand and inventory policy which can be dynamically applied based upon the observed demand patterns in the transaction history. Third, the means of identifying an optimal set of inventory control parameters is through the use of simultaneous perturbation stochastic

approximation within a simulation optimization framework. Fourth, the inventory review process modeled within the multi-item inventory simulation does not assume perfect knowledge of the actual stock level for an inventory item when the replenishment decision is made. It is modeled under the assumption that humans are performing the inventory review and, for the most part, people can not count very accurately (Kang and Gershwin, 2005). Finally, the cost function used to evaluate the performance of the inventory policies and their parameters includes a component that penalizes for causing skilled maintenance workers to be idle waiting for inventory items.

The best design of resupply networks focused on the optimal allocation of inventories within service part supply chains is of unquestionable importance to the economical maintenance of equipment (Muckstadt, 2005). Specifically with regard to service part or maintenance type inventories, as a group the repairable items comprise the largest part of the [U.S. Air Force] spares budget; in 1990 the Air Force had over $31 billion invested in repairables (Sherbrooke, 2004). With the increasing complexity of major operational systems, the increasing cost of designing and producing new systems and the inevitable decline of many raw materials, the continued, effective maintenance of currently operating major systems holds great value and importance. In addition, as a greater variety of complex maintainable systems enter the consumer market, the requirement for effective, low cost maintenance services increases. Customers have become more demanding and require customized products delivered in a consistently timely manner. As competition intensifies, product shortages and stock-outs significantly affect

companies' reputations (Paschalidis et al, 2004). Service part inventories and the tasks they support are also sensitive to the negative impact of inventory stock-outs and the delay they induce in the maintenance process. Even though the U.S. Air Force spent an average of $8.5 million per month in CY 2004 for bench stock inventory items managed under the Industrial Prime Vendor contract, the systems being maintained are multi-billion dollar fleets of highly complex, highly integrated systems maintained by skilled maintenance artisans. The lack of a single relatively low-cost inventory item is capable of creating a "work stoppage" on the aircraft depot maintenance line, idling hundreds of workers and reducing the operational capability of the fleet.

Dissertation Organization

This dissertation is organized in an attempt to be clear and logical in the presentation of this research, as would be expected. Chapter 2 starts by stating the need for a formal definition of the problem within the context of an information system and within systems in general. With the elements of the problem defined clearly, the chapter proceeds to describe the details of the problem space I am researching. Chapter 3 examines related research in the area of inventory management and simulation used within inventory management evaluation and optimization. Chapter 4 focuses on data mining research applied to an inventory management system and presents research conducted on the repair service inventory which is the subject of this dissertation. It is provided to support the efficacy of data mining applied to repair service inventory transaction sets. In Chapter 5, I address the problem of forecasting demand in the service part or bench stock inventory

environment and present research in the use of simulation to generate archetypal demand patterns and then subsequent data mining to discover these demand pattern archetypes in actual inventory transactions. Chapter 6 introduces the use of simultaneous perturbation stochastic approximation (SPSA) as a method of performing optimization via simulation of the inventory processes under examination. The cost function for the optimization is defined and the specific implementation details of the SPSA algorithm integrated with the AWESIM simulation tool are described. With the problem, tools and methods described, Chapter 7 details the experimentation and the application of the SPSA algorithm on actual service part inventory transactions which have been classified by demand type. Finally, chapter 8 closes with conclusions drawn from the research and suggests future avenues of exploration.

Chapter 2

Formal System Definition

In order for an inventory system to be optimized it requires an information system. Because a person cannot view or envision all of the details surrounding the transactions of multiple inventory items, it requires a computer and inventory information system to organize and manage the data. Given this information system, algorithms and procedures designed to manipulate the large volumes of data can be exploited such that inventory policies can be more responsive and demand forecasting more effective. So, to begin, the definition of an inventory item must be understood within the framework of an information system, and also in the same regard the demand processes and how they impact that inventory item need to be defined within an information system. In doing so, we can clearly define, and then illuminate the definitions of inventory items within the inventory itself, the policies that are used to manage those items and the forecasting algorithms used to forecast demand; and thereby set reorder levels for the items, reorder quantities and review periods.

The information system gives us the ability to view highly detailed inventory transaction history. Without the ability to view the transaction history and process it as a time series, we are not able to envision the patterns that may exist or, equally important, may not exist in the transaction history. In addition, an inventory manager would need to rely on his or her experience and intuition to determine what inventory levels are to be set, and what items should be ordered, when they should be ordered and what quantities. The basic and most rudimentary inventory system would rely strictly on the experience of inventory manager's awareness of the inventory levels

and demand patterns associated with each inventory item. This is perhaps a solvable problem by a person with a good deal of experience managing a small inventory, however when the number of inventory items exceeds a small number this task becomes impossible for the inventory manager. The number of inventory items in a typical large maintenance facility, such as an aircraft maintenance depot, exceeds 100,000. So managing over 100,000 different inventory items with different demand patterns requires more processing capability, more automated experience you might say, than is possible within the normal human.

Each of the complex systems maintained by large repair facilities is composed of several components and assemblies, each of varying complexity. For the purpose of clearer exposition the following definitions for maintainable systems and components are provided. The definitions annotated by an asterisk (*) are taken directly from (Wand and Weber, 1990) which, in turn, are taken or adapted from (Bunge, 1977 & 1979). They are used here to support the coupling and dependence that exists within complex, maintainable systems.

> The elementary notion of this formalism is a thing. All objects are things, but only some types of things are objects. Let us start with a definition of the state space of a thing.
>
> ***Definition 1****: Let X be a thing modeled by a ***functional schema*** $X_m = \langle M, \tilde{F} \rangle$, and let each component of the function
>
> $$\tilde{F} = \langle F_1, ..., F_n \rangle : M \rightarrow V_1 \otimes ... \otimes V_n$$
>
> represent a ***property*** of X. Then $F_i, 1 \leq i \leq n$, is called the i[th] ***state function*** (variable) of X, $\tilde{F}$ is called the ***total state function*** of X, V_i is the set of all possible values of the i[th] state and
>
> $$S(X) = \{ \langle x_i, ...x_n \rangle \in V_1 \otimes ... \otimes V_n \mid x_i = F_i(M) \}$$
>
> is called the ***possible state space*** of X.
>
> ***Definition 2****: Let X be a thing modeled by a functional schema $X_m = \langle M, \tilde{F} \rangle$, let $t \in M, t > 0$ be a time instant. Then a ***history*** of X is the

set of ordered pairs, $h(X) = \left\{\left\langle t, \tilde{F}(t)\right\rangle\right\}$. In turn, the notion of a history allows us to determine when two things are bonded or coupled to each other. Intuitively, if two things are independent of each other, they will have independent histories. If they are coupled in some way, however, at least one of the things' histories will depend upon the other thing's history. Thus we have Definitions 3 and 4.

Definition 3*: A thing X acts on a thing Y, denoted $X \rhd Y$ if $h(Y \mid X) \neq h(Y)$. If the history of *Y* in the presence or influence of *X* differs from the history of *Y* without regard for, or independent of *X*, then *X* acts on *Y*.

Definition 4*: Two things X and Y are coupled denoted $B(X,Y)$, iff $(X \rhd Y) \vee (Y \rhd X)$: X and Y are coupled if and only if *X* acts on *Y, or Y* acts on *X*.

Definition 5*: Let C be a set of things, where X and Y are things in this set *and* X and Y are coupled: $B_C = \{(X,Y) \mid X, Y \in C \wedge B(X,Y)\}$. Let $\sigma(C, B_C)$ be a graph, where C is the set of vertices (things) and B_C is the set of edges (couplings). Then $\sigma(C, B_C)$ is a system iff it is a connected graph.

Definition 6**:** *A maintainable, dependent, unifunction assembly* (***A***) *is the basic maintainable object with attributes defining its construction via a bill of materials (M_A) and a set of required assembly operations (θ_A), operational hours (h_A), scheduled maintenance operational hour threshold (c_A), an expected lifetime L_A, mean time to repair ($MTTR_A$), and mean time between failures ($MTBF_A$): **A** = (M_A, θ_A, h_A, c_A, $MTTR_A$, $MTBF_A$).*

Definition 7**:** *A maintainable, dependent, multifunction component* (***q***) *is a set of one or more coupled maintainable assemblies* $m_A = \{\alpha_i \mid \alpha \in A, B(\alpha_i, \alpha_j), i >= 1, j >= 1, i \neq j\}$, *a bill of materials ($M_q$) and a set of required assembly operations (θ_q), operational hours (h_q), scheduled maintenance operational hour threshold (c_q), an expected lifetime L_q, mean time to repair ($MTTR_q$), and mean time between failures ($MTBF_q$):*

q *= (m_A, M_q, θ_q, h_q, c_q, L_q, $MTTR_q$, $MTBF_q$).*

Definition 8**:** *A maintainable, independent, multifunction system* (***σ***) *is a set of one or more coupled maintainable components or assemblies* $s_\sigma = \left\{\varphi_i \mid \varphi \in \{A \cup q\}, B(\varphi_i, \varphi_j), i >= 1, j >= 1, i \neq j\right\}$*, a bill of materials* (M_σ) *and a set of required assembly operations* (θ_σ)*, operational hours* (h_σ)*, scheduled maintenance operational hour threshold* (c_σ)*, an expected lifetime* L_σ*, mean time to repair* ($MTTR_\sigma$)*, and mean time between failures* ($MTBF_\sigma$)*:* $\boldsymbol{\sigma} = (s_\sigma, M_\sigma, \theta_\sigma, h_\sigma, c_\sigma, L_\sigma, MTTR_\sigma, MTBF_\sigma)$.

Definition 9**:** *A bench stock repair service inventory item (P), is a low cost, dependent, unifunction item appearing on one or more maintenance objects' bills of material:* $\rho_P \in \left\{M_A \cup M_q \cup M_\sigma\right\}$, and at least one of the following three statements $B\left(\rho_P, A \mid \rho_P \in M_A\right), B\left(\rho_P, q \mid \rho_P \in M_q\right), B\left(\rho_P, \sigma \mid \rho_P \in M_\sigma\right)$ must be true. Note also, any of the following statements could be true: $\left\{M_A \cap M_q\right\} \neq \varnothing, \left\{M_A \cap M_\sigma\right\} \neq \varnothing, \left\{M_q \cap M_\sigma\right\} \neq \varnothing$.

Having a clear definition of a maintainable system, allows us begin to understand the complexity of the processes generating demand for bench stock inventory items. For an example of the application of these definitions; the maintainable system could be an aircraft, a component could be the landing gear, and an assembly, the brakes. The bench stock items in this example would be the bolts, nuts, washers, spacers, rivets, etc. used in the maintenance and assembly operations (θ_A, θ_q, θ_σ) of the brakes, landing gear and aircraft. (Muckstadt, 2005) and (Sherbrooke, 2004) both address the difficulty of defining an optimal method of providing support for maintainable components, ***q***. The problem examined by this

research is that of providing optimal support for the bench stock inventory processes supporting the maintainable assembly, ***A*****,** up through the maintainable components, ***q***, to the final level: the maintainable system, ***σ***.

Repair Service Inventories

A repair service inventory contains multiple stock keeping units (SKUs) in

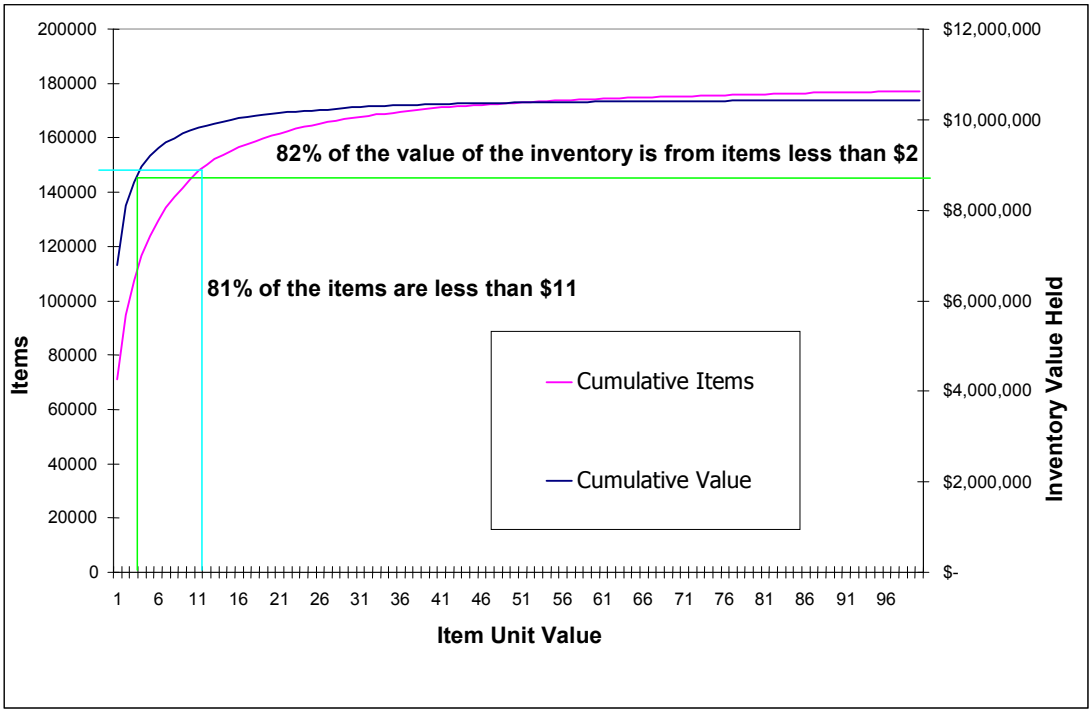


Figure 1: Bench stock inventory value

varying quantities with varying individual item values. Individual items may vary from as little as $0.001 to $10,000. The inventory may also be comprised of many thousand distinct parts supplied from a variety of vendors, each with their own replenishment lead time. Examined strictly from a monetary point of view, the bulk (82%) of the value of one specific aircraft bench stock inventory comes from individual items costing less than $2.00 per item. Figure 1 displays data, from a

snapshot in time in 2004, taken from the bench stock inventory management information system administered by a government contractor under the Industrial Prime Vendor (IPV) contract; a Defence Logistics Agency (DLA) contract for the maintenance bench stock inventory support of several U.S. Air Force aircraft.

However, management of the bench stock inventory cannot focus entirely on the monetary value of the items. Each item is carried in the inventory because it is required for the completion of one or more maintenance tasks. Bench stock inventory is also referred to within the U.S. Navy as pre-expended bins (PEB), meaning that they have already been expensed and are already assumed to be of valuable use for the maintenance process that will inevitably occur. An aspect which is assumed throughout this research is that the repair service or bench stock items being managed are not held for sale individually and are not managed as a source of income for the organization. When speaking of inventory items in this class, I am referring to items which support the maintenance repair service which is the key product or output of a large maintenance organization.

Knowledge of the method in which bench stock items are used within the maintenance cycle is important toward the understanding of both the profile of the direct item demand and the application of the inventory control policies. The bench stock items used in the aircraft maintenance facilities within the U.S. Air Force and the U.S. Navy repair depots are held in storage near the maintenance areas. Often a cabinet with multiple drawers, each subdivided into many compartments, is used to store the various parts used in the maintenance activities. This miniature "warehouse" is the site of all bench stock inventory issues and replenishments. When

an item is required for a task, the skilled maintenance worker removes the quantity of the item from its storage location. The number removed from the bin, most often, is not recorded. Therefore, this evidence of actual demand is lost. The maintenance workers act on a variety of tasks throughout the day per a job schedule. Therefore, it is difficult to tie an inventory withdrawal to a specific maintenance task.

$D = \{\gamma_1, \gamma_2, ..., \gamma_W\}$,	$P = \{(p, u_p, u_c, u_s, LT_\mu, LT_\sigma)_{\mathrm{P}}\}$,
W is the number of maintenance workers,	P is a finite number of distinct parts,
$\gamma_W \in J$	p identifies the part,
$J = \{(j, \tau_1, \tau_2, ..., \tau_C)_{\Gamma}\}$,	$u_p, u_c, u_s \in \mathbb{R}^+$, the part's unit price, cost and salvage
Γ is a finite number of distinct jobs,	LT_μ the average lead time,
j identifies the job,	LT_σ the standard deviation of lead time,
C is the number of tasks per job,	$W = \{\beta_1, \beta_2, ..., \beta_U\}$, a warehouse where a bin $\beta_U \in B$
$\tau_C \in T$	$B = \{(b, \rho, s, Q, H, O, X)_{\mathrm{B}}\}$,
$T = \{(t, k, \theta, d)_{\mathrm{T}}\}$,	B is a finite number of distinct stock locations,
T is a finite number of distinct tasks,	b identifies the stock bin,
t identifies the task,	$\rho \in P$,
$k \in K$,	$s, Q, H, O, X \in \mathbb{N}^+$
$\theta_T \in \{\theta_A \cup \theta_q \cup \theta_\sigma\}$, assembly operations	s represents the reorder point
$d \in \mathbb{R}^+$ duration measured in days	Q is the order quantity
$K = \{(k, n, (\rho_1, r_1), (\rho_2, r_2), ..., (\rho_n, r_n))_{\mathrm{K}}\}$,	H is the on hand balance
K is a finite number of distinct kits,	O is the amount on order
k identifies the kit,	X is the amount on backorder
n is the number of distinct parts in the kit,	
$\rho_n \in P$,	
$r \in \mathbb{N}^+$ required number of part ρ_i in a kit	

Table 1: Maintenance process elements

A diagram of a typical work process is displayed in figure 2. The key things to notice, relating to the inventory, are what actions take place when the required parts are not available. The artisan has few choices; go looking for the part in a secondary storage location, start on another task or wait for the item to show up in the inventory. All of these activities delay the maintenance process and increase the amount of time the maintained system is not available for operation. With the

definitions given in table 1, the connection between the maintainable system, the work schedule and the repair service inventory will become clearer.

Periodically, a work schedule, *D*, is produced that allocates the known maintenance work to the available workers. Each artisan is given a job, *J*, to complete within an estimated period of time based upon the complexity of the tasks,

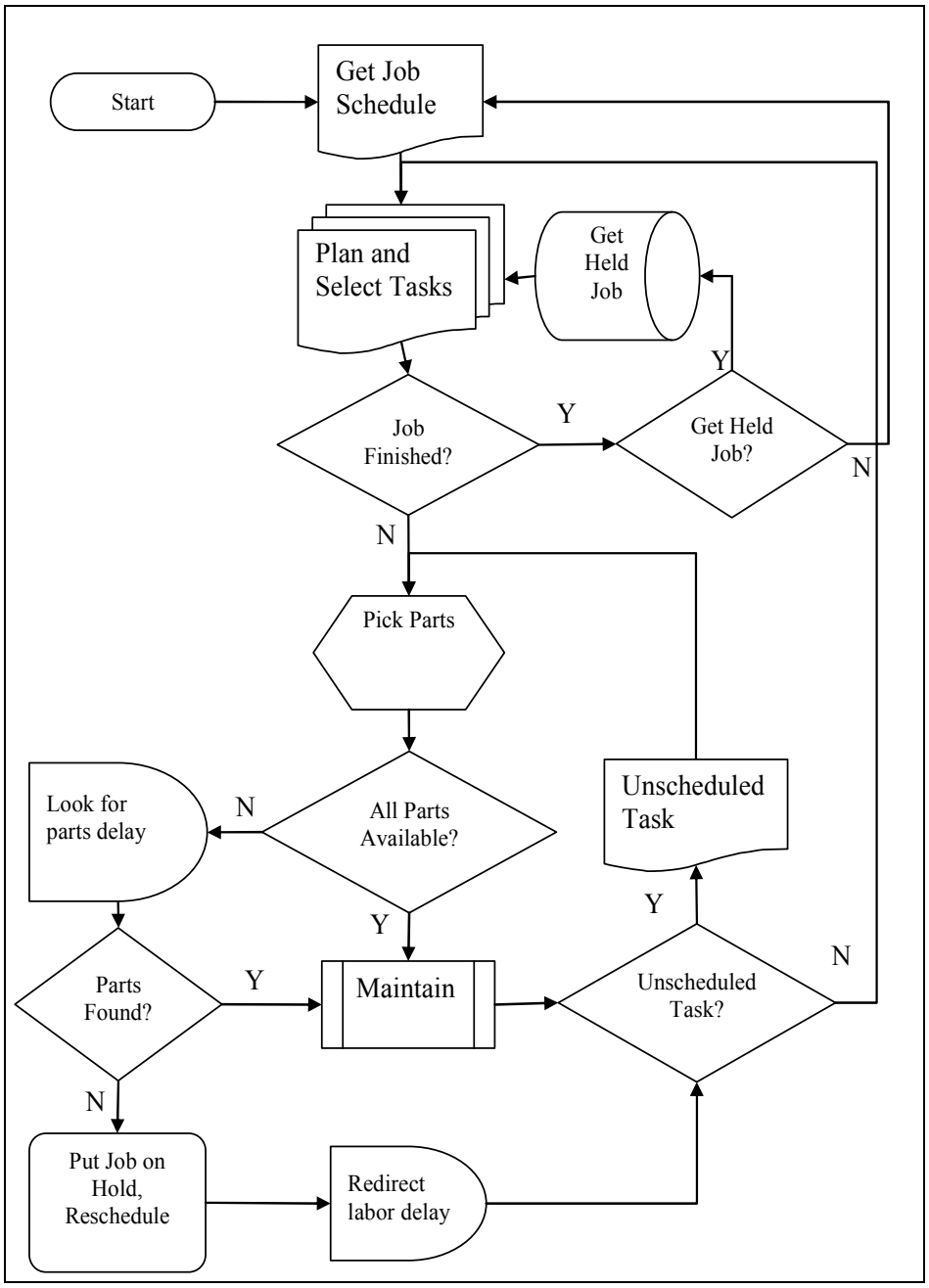


Figure 2: Maintenance work flow

T, included in J. Given the tasks within the job, the maintenance and assembly operations, θ_T, require a specific order and a set number of bench stock parts defined within a kit, K. The task, T, will require all the parts defined within K in order to complete the task. When a specific task is in work, the artisan goes to the bench stock warehouse location, W, to gather the parts required. Each of the bench stock parts, $\rho \in P$, will normally be stored in one of the bins, β, within W. Of course, this is where the maintenance process meets the inventory process. Using a given control policy and a set period, inventory workers review the contents of all the bench stock warehouse locations, W. A manual review is required because the various withdrawals from the bins, β, are not recorded in a system. Also, the number and variety of parts taken from the bins cannot be directly related to a specific kit, K, because several of the bench stock parts are common to multiple maintenance and assembly operations.

Inventory Item Behavior

Inventory item behavior can be defined through the item's various states, state transitions, and the rate at which these transitions occur when viewed from the position of the inventory bin, β, within the bench stock warehouse, W. The states in which β can exist are a supply state, a demand state, an expedite state and an idle state. In a supply state, the inventory level is at a sufficient quantity to support the maintenance operation or repair service function, and the quantity of ρ in β is above a threshold, s, such that, if demand continues at the current rate, then a replenishment order will be able to supply more parts without interrupting the operation. Another aspect of the supply state is evidence of usage. Again assuming there is evidence of

inventory usage, an inventory item location β can be considered in a demand state, if the item is on order. The presence of an order indicates the inventory position is now below s and is beginning to use any safety stock that exists in inventory that has been set aside to account for the amount of time required to replenish the inventory item. When the evidence of usage indicates that the inventory in the bin is insufficient to support demand, β transitions to the expedite state and will require additional inventory management attention. If no evidence of demand is present for a given inventory location, it is considered in an idle state.

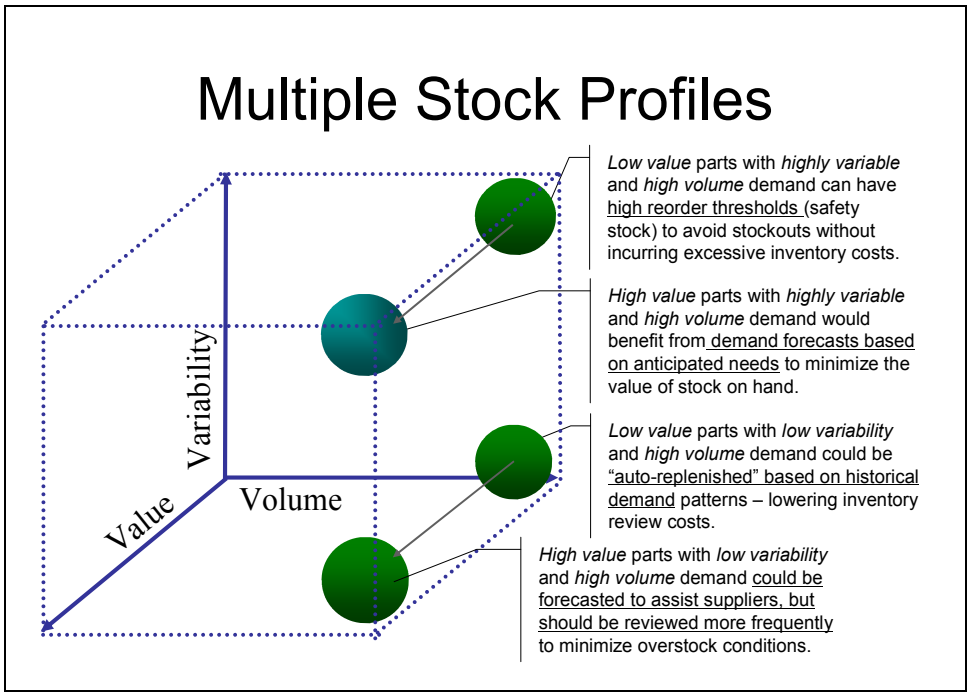


Figure 3: Multiple stock profiles

What influences the transitions from state to state and therefore, in a sense, defines inventory item behavior? Factors that contribute to inventory item behavior are the processes that trigger demand, the cost of the item, the availability of the item in the commercial environment, the complexity of the inventory item, the number of repair or maintenance tasks for which the item is required (commonality), the length

of time it takes to produce the item and the transportation environment in which this inventory exists. Because each item within the repair service inventory may vary widely, the policy to manage each, intuitively, may be different. Figure 3 presents some possible variations of the inventory management based upon the behavior "profile" of the inventory item.

Efficiently managing the behavior of the inventory has been a subject of research for many years. The classical economic order quantity, EOQ, was first introduced by (Harris, 1913) while working at the Westinghouse Corporation. Most probably, effectively controlling inventory has been a topic of serious management for hundreds, if not thousands, of years.

Inventory management models can be divided into two broad categories: deterministic and stochastic. In a deterministic model the prices, lead times, and demand are known a priori and the key is to balance the various costs such as ordering and holding costs. In a stochastic model the data points are not all known and each may have varying levels of uncertainty attached to it. These models are the most appropriate in the bench stock inventory management environment.

The first stochastic model used within the simulation experiment is the classical newsboy or news vendor problem. This is characterized by the need to make a decision about how much of an item to purchase for the next period of demand. It is a single period model and is focused on optimizing the order quantity given some knowledge of the demand in the past, the selling price (u_p), item cost (u_c), and salvage value (u_s) where $u_p > u_c > u_s$. The structure of the problem assumes that there are no fixed reordering costs and no initial inventory. The demand D is

modeled as a random variable with a continuous distribution function $F(\cdot)$. The idea is to select *y* such that the cost function *C*(*y*) is minimized where

$$C(y) = u_c y - u_p \int_0^\infty \min(y, D) dF(D) - u_s \int_0^\infty \max(0, y - D) dF(D) \quad for \;\; y \geq 0$$

Given that $\int_0^\infty \min(y, D) dF(D) = \int_0^y D dF(D) + y \int_y^\infty D dF(D)$

$$= u_c y - u_p \left(\int_{D=0}^y D dF(D) + y \int_{D=y}^\infty D dF(D) \right) - u_s \int_{D=0}^y (y - D) dF(D)$$

adding and subtracting $u_p \int_{D=y}^\infty D dF(D)$, we get

$$= u_c y - u_p \int_{D=0}^\infty D dF(D) - u_p \int_{D=y}^\infty (y - D) dF(D) - u_s \int_{D=0}^y (y - D) dF(D)$$

taking the derivative of $C(y)$ with respect to y and using the Leibnitz rule yields the first order optimality condition:

$u_c - u_p (1 - F(y)) - u_s F(y) = 0,$ where $F(y)$ is the probability $\Pr\{D \leq y\}$

therefore the optimal order quantity Q satisfies the following condition:

$$\Pr\{D \leq Q\} = \frac{u_p - u_c}{u_p - u_s}$$

This method is not used specifically within the simulation to calculate order quantities, because the period to period demand distribution is very difficult to characterize, but it is used to set an upper bound on the stocking level and reorder point. The upper bound of the stock level in any repair service bin is set to a quantity equal to the 4 month single period optimal order quantity indicated using the newsboy model. Because we do not always know the markup or the salvage of the items being stocked, a set 12% markup is applied to the unit cost and the salvage value is assumed to be 1% of the unit cost.

The classes of inventory management policies evaluated within the framework of the simulation and the experimentation are the fixed order quantity policy (r,Q), the periodic review policy (P) and the order up to policy (s,S). In the fixed order quantity model, the inventory item is stocked with a set reorder point (r) and when the

inventory position decreases below this point an order is generated for the fixed quantity (Q). Often the EOQ formula, $Q = \sqrt{\frac{2kd}{hu_c}}$, is used to set the order quantity for each bin where *k* is the order setup cost, *d* the demand and *h* the holding cost adjusted to the amount of time between inventory reviews. The reorder point in this model is used to control the level of safety stock and replenishment lead time stock. A variation of the fixed order quantity policy which is also used in the simulation and experimentation is a modified two-bin policy. The two-bin policy does not require a demand forecast, but requires a good estimate of the replenishment lead time. In this policy, two bins are designated to hold every inventory item. Inventory is drawn from one bin until it is empty, an order is placed to fill the bin, and the second bin is used to fill demand until it is empty. This, of course, assumes that the order for the first bin is received and binned before the second bin becomes empty. The variation of this policy modeled in the simulation, and the policy employed throughout much of the U.S. Air Force and Navy bench stock inventory, regards a change in the order amount. Instead of ordering the amount to fill one bin, an order is placed to fill both bins when the first becomes empty.

The periodic review policy operates under the model where the inventory is reviewed on a set interval and the order quantity is determined as a difference between a desired stocking level *S* and the current inventory position at the time of the review. The period can be determined using a technique similar to the EOQ such that the review interval, *T*, is set $T = \sqrt{\frac{2k}{hd}}$. The order up to level *S* is set in a manner analogous to setting the reorder point in the fixed order quantity method.

The order up to (s,S) policy borrows the reorder point concept from the fixed order quantity policy and the order quantity from the periodic review policy. On a given review cycle, *T*, the inventory position is examined. If the inventory position of the item is at or below the reorder point *s*, then the difference between the inventory position at the time of the review and the desired stock level *S* is the quantity ordered. Contrary to the fixed order quantity and the periodic review policies, setting the parameters for the order up to policy, (*s*, *S*, and *T*), is difficult to determine analytically. Because of this, simulation is often used to determine these values (Ghiani et al., 2004). The variations of the (s,S) policy also analyzed are the continuous review and periodic review base stock policies where the reorder point is set to *S*-1, which in effect, causes an order to be placed for the amount of the last withdrawal from the inventory bin. This policy is a critical feature of the inventory control analysis and theory presented for managing repairables by both (Sherbrooke, 2004) and (Muckstadt, 2005). In fact, in his work, Muckstadt, provides a proof of the optimality of the (*s*-1,*S*) inventory control policy, again specifically focused on the management of repairable components, ***q***.

All of these policies seek to balance the costs of holding inventory and the penalty of not stocking enough to cover demand, but all are policies that react in the inventory demand environment in which they operate. Understanding the demand process volume and variability is of paramount importance when managing inventory. Examining the complex demand process environment in which the repair service inventory operates is the subject of chapter 5.

Chapter 3

Related Research

The body of research literature dedicated to addressing the problems of inventory management is extensive and spans several years. It is quite common to find research seeking optimal solutions for managing production inventory, retail or wholesale inventory; however, the paucity of research focused on repair service inventory management is quite evident. One work (Berman et al., 1993) directly addresses inventory management supporting a repair service facility, such as an automobile body shop. They note that the assumptions supporting the demand rate for inventory used for production or manufacturing are not necessarily realistic in the service facility. The work focuses on identifying the optimal order quantity where the cost model depends upon the ordering rate, the average inventory in the system and the average number of customers waiting in queue. A simulation study is described which uses both constant and Poisson demand distributions and the research concludes that an optimal inventory policy in the presence of fixed service capacity can be found. A concluding remark states that when efficiency has a cost, it is "reasonable to inflate inventories to balance the competing costs of customer waiting, inventory holding, setup and service capacity."

The area of spare part support for processing and fabrication equipment in a semiconductor fabrication facility was analyzed by (Akcali et al., 1997). Their purpose was to find an optimal inventory policy for the complex machinery spare part inventory in order to minimize the occurrence of long-duration, unpredicted equipment downtime. One of the key factors causing these delays was equipment

downtime awaiting spare parts for repair. Through the use of simulation on four different spare parts, the experimental results, though not optimal in all cases, pointed to the application of either the continuous or the periodic order up to (s,S) management policy. During analysis of the demand patterns for the spare parts, they found highly sporadic demand in conjunction with rapidly shifting technologies and product mixes making characterizing the distribution of the demand from historical records very difficult.

Using simulation to model inventory problems has been applied by several researchers with objectives of producing either optimal or adaptive inventory control policies. (Kim et al., 2005) used simulation to test the adaptability of an error correcting inventory control policy. Although this work modeled a JIT system with an adaptive policy, the assumptions concerning the low variability of the demand make the results less useful, especially in the high variance demand environment of bench stock inventory. In their review of system dynamics modeling in supply chains, (Angerhofer and Angelides, 2000) present examples of inventory management simulation research intent on modeling for theory building and problem solving, but none with the goal of putting the research into practice. A two-echelon aircraft spare parts inventory problem was examined by (Lee et al., 2005), within the framework of a multi-objective simulation optimization. They addressed the problem of determining how many spare parts to store at a set number of airports for the purpose of increasing commercial aircraft operational availability. Not surprisingly, they expand on Sherbrooke's Multi-item technique for recoverable item control (METRIC) by identifying an optimal spare part location mix. They stated that

because the features of problems of this nature often make them mathematically intractable and to avoid too many simplifying assumptions, a simulation which modeled the problem environment was preferred over analytical methods. (Kapuscinski and Tayur, 1998) study a capacitated production inventory having periodic non-stationary demand with the use of simulation and infinitesimal perturbation analysis (IPA). Using this application of simulation optimization, they conclude that a capacity bounded order up to inventory policy is optimal for the finite horizon, the discounted infinite horizon and the infinite horizon average cost criteria.

Research closely related in structure to the work presented here was a collaboration between the University of Leeds, Leeds, UK and a small UK chemical company (Garcia-Flores et al., 2003). The academic team's method of analysis was to identify the system characteristics, classify the inventory according to demand classes, select a forecasting and inventory control policy that matched the demand class and the test the validity the solution using simulation. They reported very good anticipated cost savings for the company due to the effective coordination of the demand classification and the control policies.

Two works provide the most authoritative treatment of the inventory problem related to maintaining complex systems. The first is the work by (Sherbrooke, 2004) which presents an analysis of the maintenance repair service process at the component level, focusing on optimizing the operational availability of the system or systems. He builds his inventory model theory on the optimality of the base stock (S-1,S) control policy and upon Palm's (1938) theorem (infinite channel queuing assumption).

> Palm's theorem states that if demand for an item is a Poisson process with an annual mean $\overline{d}$ and the repair time for each failed component is independently and identically distributed according to any distribution with mean T years, then the steady-state probability distribution for the number of components in repair has a Poisson distribution with mean $\overline{d}T$.

The focus on optimizing availability rests on the foundation of this control policy such that the minimization of the component backorders is equivalent to maximizing availability. He also notes that only the repairable, component level is addressed and that the mathematical problem is more difficult when finding the optimum control policy parameters for the lower indentured components, assemblies and bench stock. The second work is by (Muckstadt, 2005) which also addresses maintenance repair service at the component level. He explores the phenomena that the demand processes for repairable components are not necessarily Poisson, and provides extensions of Palm's theorem to the nonstationary and nonstationary compound Poisson process cases.

Chapter 4

Data Mining Inventory Transaction Streams

A growing body of research is revealing the economic value and analytical potential of mining vast stores of information. The application of data mining methods to problems involving the discovery or prediction of rare events and patterns has shown promising and surprising results in areas as diverse as predicting tornadoes, electrical power consumption, customer retention, loan default and bank failure (Brierley and Batty 1999, Piramuthu 1999, Wai-Ho et al. 2003, Trafalis et al. 2005). The area of supply chain and inventory management with its constant challenges of determining how much inventory exists in the warehouse, when to order items, how much to order and how to measure and forecast demand for an item, provides an opportunity to explore the ability of data mining to help answer these questions.

One of the critical, yet often underestimated, tasks of building an effective data mining application must be performed before any actual data mining algorithms are exercised. The preparation of the initial data sets for training the mining algorithms, and the test and validation sets used to examine and support the application of the data mining model is time intensive and critical. One of the key steps in the data preparation processes is the process of feature or attribute selection (Liu and Yu 2005, Howard and Rayward-Smith 1999). This paper presents the application of a method for performing and analyzing feature selection and data mining techniques against a data set comprised of over 2 million inventory transactions, collected over five years, in support of the management of a bench stock

inventory containing over 100,000 distinct items, P, stored in over 400,000 stock locations, β. Despite the challenging lack of critical inventory planning data elements, causal attributes and key transaction metrics, a high fidelity set of attributes is sought for use within proven data mining algorithms. The primary focus and motivation for analysis is the systematic discovery of this set of attributes, drawn from a temporally normalized inventory transaction time-series, providing usable leading indicators toward predicting the rare inventory condition of an empty stock location.

The problem of feature or attribute selection, a key technique for engineering input data, is simply stated: given a set of measurements on p variables, what is the best subset of size d, such that those d variables contribute the most to discrimination. Manipulating input data sets containing multiple attributes in order to reduce the number of dimensions is done for a variety of reasons including easier subsequent analysis, improved classification performance through a more stable representation, and removal of redundant or irrelevant information (Webb 1999). The search for an optimal solution to this usually intractable problem has led to a proliferation of feature selection algorithms, but has not brought about a general methodology for intelligent selection from the growing list of algorithms (Liu and Yu 2005). Among the research reviewed which applied data mining methods in order to discover an infrequently occurring critical event or pattern, no single method or common approach appeared to be used by the researchers (Brause *et al.* 1999, Dhond *et al.* 2000, Vilalta and Ma 2002, Yohda *et al.* 2002, Wai-Ho *et al.* 2003, Wilson *et al.* 2003, Trafalis *et al.* 2005). However, evident within this body of research is the

importance of domain knowledge of the data, and a careful selection of the data preparation and mining algorithms applied. Some effort has been made to provide initial benchmarking techniques for attribute selection, and this research concluded that like learning algorithms, there is no single best approach for all situations. What is needed by the data miner is not only an understanding of how different attribute selection techniques work, but also the strengths and weaknesses of the target learning algorithm, along with background knowledge about the data if available (Hall and Holmes 2003).

Through experimentation into improving the prediction of mid- and long-term weather forecasts, researchers found information gain to be a good indicator of important features (Howard and Rayward-Smith 1999). An analysis of the impact of feature selection was conducted on two "rare event" discovery problems, identifying loan default-prone customers and predicting bank failures. This analysis described improving feature selection through the application of the Hausdorff distance measure and found the classification accuracy of the decision tree algorithm, after pre-processing through the Hausdorff distance measure filter, was the same as that generated without the pre-processing. However, the same accuracy was obtained with fewer features (Piramuthu 1999). Research which specifically targeted mining transaction data in a supply chain or inventory management environment has also shown good results revealing that, through the analysis of thousands of transactions by a neural network, a data mining implementation could result in a significant reduction of the inventory cost held by a pharmaceutical company (Dhond *et al.* 2000).

Using a combination of feature selection and input engineering approaches, (Guyon and Elisseeff 2003, Witten and Frank 2005, Liu and Yu 2005) a number of attribute sets and their effectiveness toward improving the predictive performance of data mining algorithms were examined. This set of attributes is drawn from the transaction stream gathered by an enterprise supply-chain software system during the normal management of the bench stock inventory. These transactions include the date, time and often quantity related to the specific event or action, such as orders, receipts, stock shortage notices, stock outages, and inventory location reviews. Yet, the inventory under examination, and the processes that support its management do not allow for the typical inventory metric collection. Critical features such as time of demand and quantity issued are not recorded; maintenance workers remove items from the inventory locations for immediate use in the maintenance task. This fact complicates the problem of predicting these rare events. In preliminary analysis, (Beardslee and Trafalis, 2005) established, through a series of experiments, that several data mining algorithms could provide a prediction capability with a minimum degree of confidence. Limitations of this previous work, however, were the lack of a methodical feature selection approach, no analysis of the impact of the various attributes on the performance of the data mining algorithms examined, and the treatment of the multiple time-series transaction streams as a single data set.

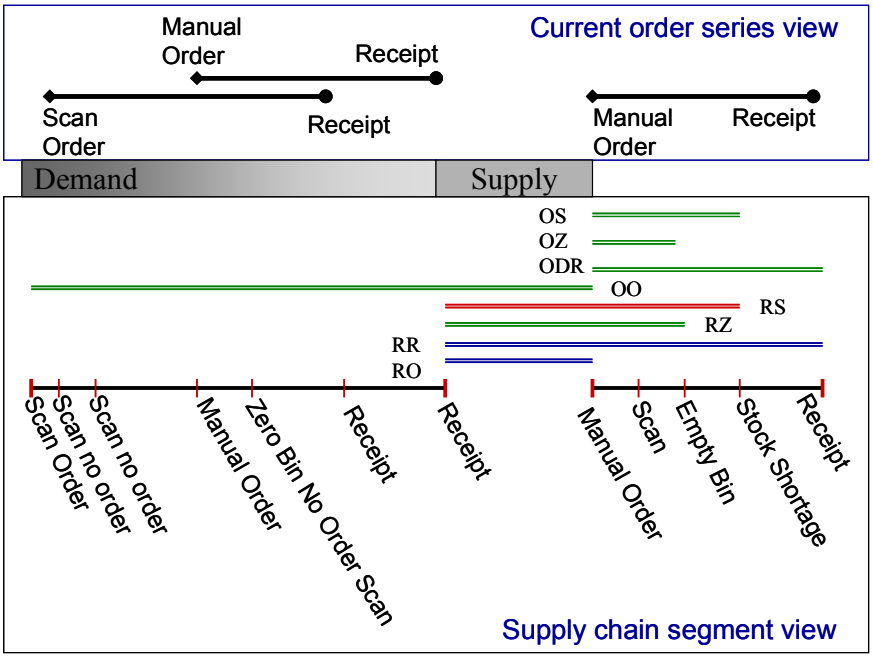


Figure 4: Supply-Chain Segment

Data Pre-Processing

Metrics are gathered and stored in an enterprise supply chain management system tracking the performance of the stock location, β, and the supply chain that provides for the replenishment of the stocked item, P. The sources of these data metrics are the inventory reviews using handheld scanners, stock outage notices, manual orders placed by inventory managers, and inventory item receipt actions. Because concurrent orders may be placed against a single inventory location, one through the scanning process and one through manual order placement, a clear, continuous time-series defining the transaction stream is difficult to construct from time contiguous orders and receipts. An accurate view of the inventory stock replenishment state can only be produced after the transactions are combined into a normalized time-series representation called a supply-chain segment (Beardslee and

Trafalis 2005). Each supply-chain segment, fig. 4, is built from eleven transaction events that provide evidence revealing the state in which the inventory stock location exists. In addition, the source orientation of this time-sensitive state information is from the stock location itself, not from the overlying order processing apparatus.

Attribute	Description
RR	Days between supply-chain segment end dates
RO	Days between supply-chain segment end and subsequent segment start
RZ	Days from segment end to a stock outage
RS	Days from segment end to a stock shortage
OO	Days from segment start to segment start
SB	Number of stock shortages in a segment
SE	Number of stock shortage resolutions in a segment
VO	Number of voided orders in a segment
NOS	Number of bin review scans that do not produce orders
OS	Days from the segment start to the first stock shortage notice
OZ	Days from the segment start to a stock outage
THIRTYDAYUSAGE	The average inventory item usage in a thirty day time period
AVG_DAYS_BETWEEN_ORDERS	Average days between order transactions
STDV_DAYS	Standard deviation of days between orders
AVG_PIPELINE_DAYS	Average number of days from order placement to receipt of goods
MEDIAN_PIPELINE_DAYS	Median number of days from order placement to receipt of goods
STDV_PL_DAYS	Standard deviation of pipeline days
N_COUNT	Number of receipts processed
EXPEDITECOUNT	Number of expedited orders
QTY_RCV_MEDIAN	Median quantity received into a stock location

Table 2: Key Attributes

On a daily basis, the metrics are collected and added to a decision support data warehouse. An extract, transformation and load (ETL) process creates the supply-chain segments critical for data mining. The key metrics describing a stock location and its set of supply-chain segments were the source for the data mining analysis that follows. These key attributes are detailed in table 2. In addition to these attributes drawn directly from the data warehouse, a set of derived data values were also used to support the stock outage prediction process. A ratio of the receipt-to-order days

(RO) to the receipt-to-receipt days (RR) describes the proportion of time the stock location acted in the supply state compared to the entire supply and demand cycle duration. A ratio of the observed usage, measured when the stock location is in the supply state, to the static re-order quantity normalized to day units (IDQ-to-ROQ) provides a numeric description of how the current "demand" matches the expected demand. A third derived attribute is the attention level, ALEVEL. It is a discretized, four-value combination of the RO-to-RR ratio, and the IDQ-to-ROQ ratio.

After the final set of data elements were chosen to be included in the data sets for analysis, extract queries were prepared to create the training and test datasets. Two separate approaches to mining the transaction series were taken. The first, *aggregate method*, treated all the stock location transaction streams as a single data source, seeking a common, inventory-item-independent attribute set which performed as a leading indicator of a stock outage. In the second approach, *individual method*, we examined each stock location transaction stream as an independent time series. The training sets for the *aggregate method* contain a sample of full day transaction sets, selected randomly from the transaction history, comprised of a total of 70,955 separate supply-chain segments. Twenty four test sets were created from randomly selected days chosen from the transaction history. Each test set contains over 2,975 records and all the test sets contain a total of 76,760 test instances.

For the *individual method*, 57 inventory locations were selected from the set of 400,000 item bins. Four years of supply-chain segments were gathered for each of these stock locations for use as training sets, a total of 4,550 segments. One year of supply-chain segments were collected for use as the test data sets, a total of 1394

segments. These transaction streams were selected based upon the number of supply chain segments available in the transaction history. The difficulty facing this second approach is one of finding transaction streams with enough data points to train a data mining algorithm. Each stock location history is defined by a unique set of transactions; however, for many items in the maintenance inventory examined, the number of state changes experienced by a given stock location may be very limited. This fact was one of the driving factors in attempting to use the transaction streams as an aggregate data source as described in the first approach.

The class value defined within the data is a nominal value (A or E) created from the discretization of the number of days from the end of the sample supply-chain segment to the next occurrence of a stock shortage or outage in the transaction history of the subject stock location. The break point for the different values is the average number of days, 15, required to effectively respond to a stock outage.

Method of Analysis

With the set of attributes and class defined, and the initial training and test data sets created, the task of selecting the subsets of these features and evaluating these subsets against the chosen data mining algorithms follows. This process was aided by the application of a *unifying platform* concept defined to provide guidelines toward building an integrated system for intelligent feature selection (Liu and Yu 2005). The feature subsets output from this process were then evaluated using tailored training and test sets against the Naïve Bayes, Bayesian network, C4.5 decision tree classification algorithms and the sequential minimal optimization

(SMO) algorithm for training a support vector classifier as implemented within the Weka Explorer data mining workbench, version 3.4.4 (Witten and Frank 2005).

Feature Selection

The unifying platform for intelligent feature selection describes the decision factors to be considered when approaching a data mining problem, specifically focused on the input data engineering. Eight decision dimensions are divided between the two key determining factors of knowledge and data. Currently, the knowledge factor covers *Purpose* of feature selection, *Time* concern, expected *Output Type*, and *M/N Ratio* – the ratio between the expected number of selected features *M* and the total number of features *N*. The data factor covers *Class Information*, *Feature Type*, *Quality* of data, and *N/I Ratio* – the ratio between the number of features *N* and the number of instances *I* (Liu and Yu 2005).

The *purpose* of the data mining under analysis is to produce a prediction of inventory stock outages. As such, the unifying platform recommends focusing on algorithms in the wrapper model of feature selection. There are three general approaches to feature selection. The filter method evaluates the features and selects the best subset, or ranks the features, based upon independent evaluation criteria and is performed before the data mining algorithm is introduced to the solution. The wrapper method uses a search algorithm to select the feature subsets and evaluates the performance of the feature subset using the target data mining algorithm. The hybrid method applies a combination of these methods. The *time* decision dimension was not applied to this analysis because the final implementation of the feature selection process was unknown and so was not critical. The *output type* of the features

evaluated was both minimal subsets and ranked lists. The *M/N ratio* was unknown; therefore both sequential search and random search methods were evaluated. Examining the data factor dimensions of the unifying platform; the *class information* was available, the *feature types* included both continuous and nominal values, the *quality* of the data available was good, and the *N/I ratio* was usual, i.e. the number of features was far less than the number of instances.

After evaluating these decision factors, a test of 19 feature selection approaches applied to four data mining algorithms was constructed. The feature selection methods used include seven filter and twelve wrapper approaches. Of the seven filter methods, four evaluate subsets and three evaluate individual attributes. The filter methods that evaluated individual attributes produced ranked lists of attributes. Only those attributes with an attribute ranking higher than .0025 were included in the feature subsets tested.

The first of these ranking techniques investigated was information gain. If A is an attribute and C is the class, (1) and (2) give the entropy of the class before and after observing the attribute.

$$H(C) = -\sum_{c \in C} p(c) \log_2 p(c), \tag{1}$$

$$H(C \mid A) = -\sum_{a \in A} p(a) \sum_{c \in C} p(c \mid a) \log_2 p(c \mid a). \tag{2}$$

Information gain, IG_i, is the amount of entropy decrease for class C reflecting the additional information about class C provided by attribute A_i (3) (Quinlan 1993).

$$IG_i = H(C) - H(C \mid A_i). \tag{3}$$

The second attribute ranking method evaluated was symmetric uncertainty. It is a method of measuring the correlation between two attributes *A* and *B* (4). Correlation-based feature selection determines the goodness of a set of attributes using (5), where *C* is the class attribute and the indices *i* and *j* range over all the attributes in the set (Witten and Frank 2005).

$$U(A \mid B) = 2\frac{H(A) + H(B) - H(A \mid B)}{H(A) + H(B)} \tag{4}$$

$$\sum_j U(A_j \mid C) \Big/ \sqrt{\sum_i \sum_j U(A_i \mid B_j)} \tag{5}$$

The final ranking method examined was the ReliefF ranking scheme. ReliefF works by sampling an instance from the data and then locating its nearest neighbor from the same and opposite class (in the two-class problem). The values of the attributes of the nearest neighbors are compared to the sampled instance and used to update relevance scores for each attribute. The multi-class extended ReliefF algorithm finds the nearest neighbors from each class that differ from the current sampled instance and weight their contributions by the prior probability of each class. The rationale behind the ReliefF algorithm is that useful attributes should differentiate between instances from different classes, and should have the same value for instances from the same class (Hall and Holmes 2003).

Two of the filter methods evaluated work by examining subsets of the attributes, as opposed to individual attributes, and produce sets of attributes as output. The first was the Correlation-based Feature Selection, CFS, algorithm which evaluates subsets of features based on an heuristic that takes into account the usefulness of individual features for predicting the class along with the level of

intercorrelation among them (Hall and Homes 2003). The second subset filter method was the consistency-based subset evaluator which uses the following consistency metric (Liu and Setiono, 1996), C_s, to determine if a subset of features divides the data into subsets with a strong single-class majority:

$$C_s = 1 - \frac{\sum_{i=0}^{R} |U_i| - |\mathrm{M}_i|}{N} \tag{6}$$

where s is an attribute subset, R is the number of distinct combinations of attribute values for s, $|U_i|$ is the number of occurrences of the ith attribute value combination, $|\mathrm{M}_i|$ is the cardinality of the majority class for the ith attribute value combination and N is the number of instances in the data set (Hall and Homes 2003). The consistency of any set of attributes can never improve on that of the full set, so this evaluator is usually used in conjunction with a random or exhaustive search that seeks the smallest subset whose consistency is the same as that of the full attribute set (Witten and Frank 2005).

As part of the feature selection evaluation, each of the two subset-evaluating filter methods and the two wrapper methods used two search algorithms to generate the candidate feature subsets. The first search algorithm conducts a greedy hill climbing with backtracking. The second search method uses a simple genetic algorithm (Goldberg 1999). The parameters include population size, number of generations, probabilities of crossover and mutation. Both searches were initialized with an empty set of attributes and so conducted their searches using forward selection.

Two wrapper methods for feature selection were examined using each of the two search methods identified above. Both of the wrapper algorithms,

ClassifierSubsetEval and *WrapperSubsetEval* in Weka, employ a target data mining algorithm as the feature subset evaluation method. However, *WrapperSubsetEval* also performs a cross-validation step to estimate the accuracy of the learning scheme for each attribute subset.

Data Mining

Starting with the *aggregate method*, after executing the 19 feature selection methods against the full aggregate training data set, surprisingly, 19 unique subsets of the features were identified. One training and 24 test datasets were then created using only the attribute subsets selected by each method used. Subsequently, using the *individual method*, the same set of feature selection algorithms were applied to the individual stock location baseline training data sets, however creating a set of test

Feature Selection Methods	Training Series	Attributes Names	Number of Features	Percent Correct	Kappa Statistic
13	46	ACLASS	1	89.6711	0.48883
11	96	ACLASS RZ	2	89.9419	0.50758
10	123	RZ	1	89.1863	0.47929
9	17	ACLASS IDUtoROQ RZ	3	89.0763	0.47779
8	22	ACLASS NOS RZ	3	89.9546	0.50719
6	40	ACLASS RC RR	3	89.2089	0.47842
5	37	ACLASS IDUtoROQ RC RR RZ	5	89.8275	0.50394
5	29	ACLASS IDUtoROQ NOS RO ROoverRR RR RZ	7	88.9650	0.47691
		Baseline (see Table 1)	20	88.4995	0.47068

Table 3: Selected Attribute Subsets

datasets to evaluate these results was more involved. Feature selection against the individual transaction stream training sets produced 317 distinct attribute sets of which 132 were generated by two or more feature selection methods. From this list, the top eight attribute subsets, shown in Table 3, were used to create training and test sets for the *individual method* experimentation. The training and test datasets from

both methods were then used to analyze the performance of each of the four data mining algorithms chosen. Each combination of mining algorithm and feature-selected subset data was executed 10 times using 10 fold cross-validation.

The Naïve Bayes probabilistic learner and the C4.5 decision tree algorithm (J4.8) were selected because they represent two quite different approaches to machine learning and they are relatively fast, state-of-the-art algorithms that are often used in data mining applications (Hall and Holmes 2003). The Bayesian network method of data mining was selected for two reasons. First, it represents a method which combines the strengths of the decision tree learner and the probabilistic learner through the use of directed acyclic graphs. Bayesian networks are a special case of a wider class of statistical models called graphical models, which include networks (called Markov networks) with undirected edges (Witten and Frank 2005). Second, it is very likely that the attributes which comprise the supply-chain segment contain a high degree of dependency among them, especially when viewed in the temporal aspect of the transaction metrics gathered and the maintenance process in which the items are used. Therefore, the validity of using the Naïve Bayes probabilistic learner must be considered in light of the attribute independence assumption. Fortunately, Bayesian networks help answer this concern because they allow for modelling of arbitrarily complex dependencies between attributes (Wang and Webb 2002). The fourth data mining method applied uses John C. Platt's support vector classification approach called the sequential minimal optimization (SMO) algorithm.

Results

Because the primary aim of the data mining task explored was to identify a set of features which provide an acceptable prediction of a rare event, the measures used to evaluate the feature selection and data mining approaches focused on evaluating the accuracy and precision of the predictions provided. Yet, because of the nature of infrequent occurrence of the target event within the transaction stream, even a classifier predicting a success rate over 97.9% may still misclassify all the stock outages as normal demand stock locations. To address this condition, two basic measures were used to compare the results. The first metric used was the percentage of correct classifications determined by the data mining approach, often referred to as success rate. The second metric was the *Kappa statistic* which provides a measure of the agreement between predicted and observed categorizations of a dataset, while correcting for agreement that occurs by chance (Witten and Frank 2005). The general formula for the Kappa statistic can be written as follows:

$$\mathrm{K} = \frac{P(Obs) - P(Expect)}{1 - P(Expect)} \tag{7}$$

P(Obs) is the observed proportion of true positives, TP, and true negatives, TN, and *P(Expect)* is the expected proportion of TP and TN, assuming a binomial distribution of TP and TN. Therefore, for a feature subset to be identified as better than the baseline feature set, the success rate and Kappa statistic must be greater than or equal to the baseline feature set, and contain fewer features.

The results of the separate feature selection data mining sessions under both the *aggregate method* and the *individual method* were tested using the Ryan-Einot-Gabriel-Welsch (REGWQ) multiple range test at a significance level of .05. Within

the experimentation, using both methods and the highest performing feature subsets, none of the data mining methods tested performed significantly better than any of the others.

As indicated above, the feature selection under the *aggregate method* produced a different feature subset from each feature selection algorithm. This lack of consensus among the feature selection methods was a leading indicator of the data mining results using the attribute subsets. Within the experimentation, the best success rate, 96.3662, and Kappa, 0.0868, were achieved by the baseline set of 20 attributes. Six other attribute subsets achieved results that were not significantly different from these baseline results, but none of these subsets contained less than 12 attributes. The relatively high success rate coupled with the low Kappa statistic reveals that most of the misclassifications are being taken from the minority class, in this case the class indicating a stock outage.

The feature selection experimentation using the *individual method* yielded much more promising results. Of the eight attribute subsets tested, all performed better than the baseline, however not statistically better. Yet, all of the top subsets contained less than eight attributes. The highest success rate 89.9546 was achieved using a subset of three attributes and the highest Kappa statistic, 0.50758, was returned by a subset containing two features, see Table 3. Each of these attribute subsets were identified by multiple feature selection methods acting on multiple individual transaction time-series. The top two subsets were identified by 12 of the 19 feature selection algorithms tested. The lower success rate, together with a moderately high Kappa statistic, indicates that the misclassifications are coming from

both classes. A success rate of nearly 90 and a Kappa around 0.5 would provide a predictor with adequate precision and accuracy for use by an inventory manager searching for the next impending stock outage.

Data Mining and Feature Selection Observations

Within the metrics-deprived inventory transaction data environment, discovering a means to provide inventory managers with effective decision support tools is a complex and engaging task. The research presented in this chapter shows that data mining techniques, enhanced through the application of an intelligent process of engineering the input data, can provide a method of predicting stock outages with a reasonable degree of precision and accuracy. It also shows a measured approach to feature selection, using the unifying platform for intelligent feature selection, and validates its recommendations. Using the results of this feature selection, smaller subsets of features were found that performed as well as the baseline feature set. A notable result of this research was the clear demonstration of the need for multiple performance evaluation statistics, especially when examining data mining methods seeking to predict rare events. Even though no individual feature selection algorithm stood out as the best performer within the confines of the experiment, the consensus of the feature selection methods, displayed using the *individual method* of time-series data mining, lends validation to the attribute subsets selected.

No subset of features in conjunction with the data mining algorithms could identify a common pattern among the aggregate of inventory transaction series. However, applying feature selection and data mining methods to the individual

inventory transaction streams identified several attribute subsets which provided a reasonable degree of precision and accuracy for predicting the stock outage rare event.

The research presented in this chapter shows the ability of data mining to provide a clearer view of the information that is contained within the transaction time-series created during the daily processing of repair service inventory items. This data mining is extended in the next chapter by using simulation generated training data to guide the classification algorithms. Whereas the data mining and feature selection just presented focuses on discovering which attribute of the transaction stream will yield the best indicator of a stock out, the data mining that follows seeks to classify the transaction streams at a more general level.

Chapter 5

Identifying Demand Patterns

One of the key difficulties related to service part inventories arises from the fact that much of the demand for these bench stock items is driven by numerous, inter-related factors, not only system or equipment failure. Regularly scheduled maintenance is also a source of inventory item demand. Along with scheduled maintenance, additional system problems often are identified for repair during the actual scheduled maintenance task, spawning additional demand for service parts from the inventory (see the "unscheduled task?" decision block in figure 2). Forecasting the inventory item demand generated by these interacting processes is usually an unsuccessful endeavour. This is especially evident when considering the unknown affected interrelated system components, $\boldsymbol{q}$, each requiring different (and possibly intersecting) sets of parts, $M_{\boldsymbol{q}}$, to complete each maintenance and assembly operations, $\theta_{\boldsymbol{q}}$. The combination of these factors, including non-scheduled and non-recurring events such as system recalls or periodic system upgrades, contribute to the description of a highly complex, inherently stochastic inventory management scenario. Figure 5 presents a notional graph of the probability of maintenance throughout the lifetime of a complex system, such as an aircraft. Throughout the lifetime of the system the requirement for maintenance will change, some of this change will be scheduled and some will not. Every component, assembly and bench stock inventory item which is part of the same maintainable system is, by definition, coupled to the complex system and therefore shares its maintenance history. Each lower indentured component and assembly shares the maintenance history of the

maintainable component of which it is a part. From this highly dependent structure, the assumption of an independent and identically distributed random variable as a valid representation of demand for lower indentured maintenance items is a flawed over-simplification. According to observations from research by (Muckstadt, 2005), the general assumptions of independent and identically distributed processes describing demand and resupply times are not necessarily applicable. He states that in "many circumstances both the arrival and resupply processes are time dependent" and as such the applicability of the stationary assumption is "limited to certain dynamic environments." (Sherbrooke, 1984) supports the selection of a Poisson process with non-stationary increments to model demand processes with evidence that the variance-to-mean ration for a repairable item tends to increase as the time period of measurement increases.

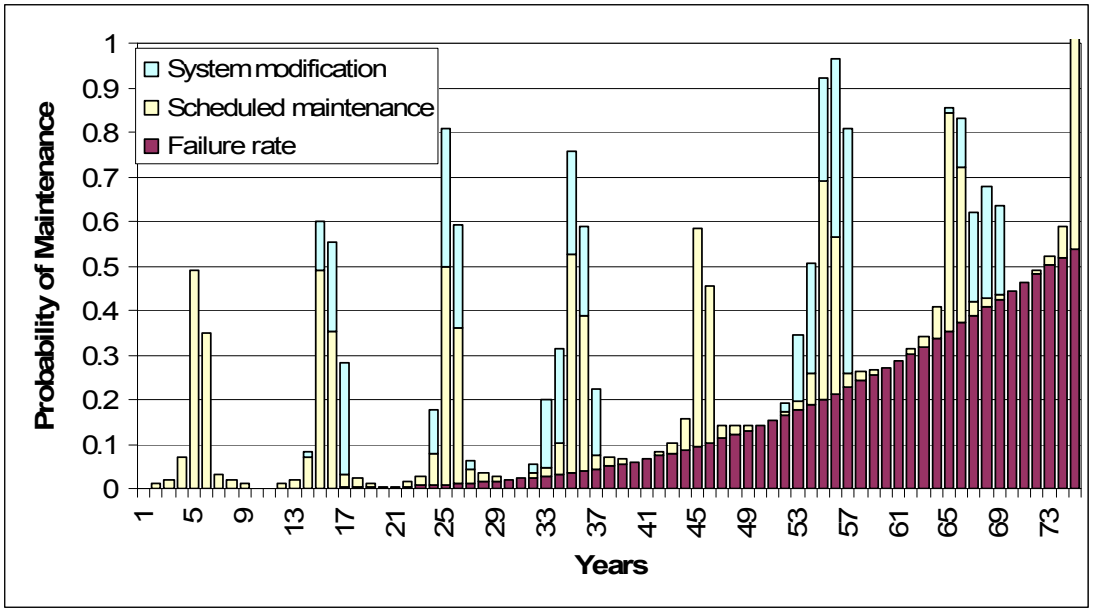


Figure 5: Demand Sources (notional data)

Because this endeavour is attempting to provide a method of optimizing service part inventory management in a stochastic environment, by default our initial

objective is a robust method of assessing demand. As such, this method must acknowledge two fundamental principles of modern operations management under uncertainty, identified by (Simchi-Levi et al., 2004), (Nahmias, 2005), (Sheffi, 2005), and (Bertsimas and Thiele, 2006): point forecasts are meaningless and should be replaced by range forecasts, and aggregate forecasts are more accurate than individual forecasts. Therefore, a significant aid to this process would be the ability to identify the type of demand structure dominating a given service repair inventory item, *P*.

Forecasting methods

Several demand forecasting methods have been developed over the years and incorporated into commercial software to help address the difficult inventory stocking and sales projection questions requiring critical answers. There are both qualitative and quantitative methods in use by nearly every business trying to effectively mange their value chain. The difficulty with the application of most current forecasting methods is that the final output of the forecast algorithm is a point forecast. In addition, short-term forecasts are more accurate than medium and long-range forecasts, as a rule. Forecasts derived using simpler methods which are easier to understand and explain are the most common, and in the business context, complex forecasting techniques rarely produce better results (Ghiani, 2004). The forecasting methods that most closely relate to the method of demand characterization described below are the time series extrapolation techniques. These include time series decomposition, the naïve approach, the moving average, exponential smoothing, auto-regressive moving averages, auto-regressive integrated moving averages and the Box-Jenkins method.

Time series decomposition

Forecasting demand in situations fraught with uncertainty requires uncovering the underlying patterns from the available information (Krajewski et al. 2007). Various terms have been used to describe the different effects that are basic patterns of demand time series. Some sources describe 4 effects or patterns and some 5, however the key concept is the same. Certain patterns can be identified in the demand time series that can be used to classify demand for more accurate forecasting.

Trend. An upward or downward trend identifiable in the demand.

Cyclical. Changes in demand caused by long cycles, such as business cycles, economic changes, changes in government spending, etc.

Seasonal. Variations in demand caused by or coinciding with seasons of the year or multiple years.

Periodic. Demand that is fluctuates with a normal period around a relatively constant mean.

Level shift. Describes demand patterns displaying a sudden positive or negative change in the mean from one relatively constant mean to another.

Residual or Random. The portion of the demand time series that defies explanation.

Combining the general idea of time series decomposition and the concept of pattern recognition inspires a natural progression to the application of transaction time series data mining. Data mining whose goal is to classify demand based upon the pattern or inherent structure found in the transaction time series. One of the first

requirements for data mining classification algorithms is a set of data which is representative of the type of actual data targeted for analysis. Many data sets have natural classification built into their definition. A set of symptoms points to a class of disease or a finite group of attributes can classify plants or animal. Unfortunately, the classification of a demand time series is not inherent, or necessarily obvious. Therefore, a method is proposed, using a simulation of the demand processes found in the repair service inventory management environment, to generate demand time series which are representative of the classes of demand expected to be encountered.

Generating Archetypal demand

Simulation description

Generating four separate types of transaction time series patterns was the purpose of the archetype demand simulation. These four types are identified as periodic, seasonal, level-shift, and sparse. Periodic demand refers to demand patterns that have a relatively regular pattern with a short demand interval. In the experimentation that follows, the transaction sets were assumed to be monthly transaction data, therefore a periodic demand pattern would be a set of demands occurring monthly with a high degree of regularity. A seasonal demand pattern is defined here as a periodic demand pattern with an interval between 3 and 12 months. This seasonality may exist concurrent with a periodic demand pattern or as a purely seasonal demand series. The level-shift demand pattern is defined as a significant positive or negative change in the regular demand pattern. Finally, the sparse demand pattern includes any demand time series that contains so few demand data points that

reasonable estimates of future demand cannot reliably be forecasted by conventional methods.

The archetype demand simulation models were created using Visual SLAM and AWESIM simulation software (Pritsker and O'Reilly, 1999). The network models for these simulations are found in appendix A. Each demand transaction stream generated represents 36 months or three years of simulated service part inventory demand. To simulate the *periodic* demand pattern, the mean and standard deviation of the demand were read from a set of sample demand series, including mean demand values ranging from 1 per month to 8500 per month with standard deviation values representing both low and high variance demand. These values were then used as the mean and standard deviation of a log normal distribution from which the mean demands of the transaction series samples were drawn. A separate sample from a log normal distribution provided the standard deviation of the demand for the transaction series. These samples of mean demand and standard deviation were then used as the mean and standard deviation in a separate log normal distribution to provide the simulated number of parts needed in a given repair operation. This number of parts needed per repair operation was then multiplied by a sample from a Poisson distribution, with a mean of 1, providing the estimate of the number of repair events in a given month. The defining aspect of the *seasonal* demand patterns was the same whether concurrent with an underlying *periodic* demand pattern or not. A seasonality or season periodicity was selected from a uniform distribution from 3 to 12. A seasonal demand delta was selected from a log normal distribution as a uniform increase or decrease of the mean demand. This positive or negative delta

was then applied to the transaction stream at the season interval. The *level-shift* demand pattern was simulated by defining a single point in the transaction stream where a positive or negative demand shift occurs. This shift point was a sample from a uniform distribution from 4 to 30. The shift delta was determined by sampling from a uniform distribution from .3 to 5.5 and multiplying this factor by the mean demand. After the demand shift point had passed in the demand series, the *level-shift* was applied by adding (or subtracting) this value to (or from) a *periodic* demand transaction stream. The final demand pattern, the *sparse* demand pattern, was generated by creating the demand mean in the same method as the *periodic* demand. Then the arrival of the demand was controlled by taking the nearest integer sample from a normal distribution with a mean of 0 and standard deviation of 8 and testing whether this value was equal to 0. If this test was true, a demand was generated, otherwise a demand of 0 was generated in the demand series.

Data Mining Methods

The archetypal training sets output from the simulation and the four test data sets were evaluated against the Naïve Bayes, Bayesian network, C4.5 decision tree classification algorithms, and the sequential minimal optimization (SMO) algorithm for training a support vector classifier as implemented within the Weka Explorer data mining workbench, version 3.4.4 (Witten and Frank 2005). These data mining algorithms were chosen because they showed promising results in the experimentation presented in Chapter 4 and for the additional reasons stated previously.

Experimentation

Eight training files were generated using the archetype demand simulation. Each simulation run used different input parameters for the mean demand and the amount of demand variation represented in the transaction streams created for the training datasets. These training files were then used for input into each of the data mining methods mentioned in the previous section.

TRAINING FILE	Instances	Periodic	Seasonal	Level	Sparse
DEMAND PATTERN TRAIN 1	16000	4809	2836	3151	5204
DEMAND PATTERN TRAIN 2	106000	33468	18824	24105	29603
DEMAND PATTERN TRAIN 3	8000	2573	1437	1871	2119
DEMAND PATTERN TRAIN 4	8000	2581	1457	1882	2080
DEMAND PATTERN TRAIN 5	8000	2576	1462	1882	2080
DEMAND PATTERN TRAIN 5 LVAR	8000	2581	1457	1882	2080
DEMAND PATTERN TRAIN 5 LRANGE	8000	2581	1457	1882	2080
DEMAND PATTERN TRAIN 7 ZSSN	10000	2581	1956	1882	3581

Table 4: Archetypal Demand Training Files

The trained classification models built with each of the data mining algorithms using 10-fold cross validation were then used to classify four test files. The four test files represent a set of hand-classified aircraft repair item inventory transactions (*H*), a set of aircraft rivet transactions "classified" using the Box-Jenkins time series analysis (*R*), a set of military tracked vehicle repair part transactions analyzed using Box-Jenkins (*V*), and a set of oil and chemical transactions analyzed with the Box-Jenkins method (*P*). Three of these test data files come from service repair part inventories. The oil and chemical demand transaction series were included to provide the initial validating support that the demand sources modeled in the simulation were service repair part items and not commodity type inventory items like oil. Therefore, the expectation was that the classifiers would perform poorly on the oil and chemical test

data. The results of the testing are presented in table 5. The Bayesian network classifier was the most effective of the four tested, followed closely by the C4.5 decision tree classifier.

	Success Rate (%)				
	Training	**Test H**	**Test R**	**Test V**	**Test P**
Bayesian Network	70.31	69.71	75.37	55.38	41.30
C4.5	68.11	66.73	73.11	54.26	37.77
SMO Linear	46.49	63.64	68.07	43.59	26.68
Naïve Bayes	41.71	52.70	57.72	44.31	26.11
	Kappa Statistic				
	Training	**Test H**	**Test R**	**Test V**	**Test P**
Bayesian Network	0.59	0.44	0.43	0.35	0.18
C4.5	0.57	0.38	0.38	0.34	0.17
SMO Linear	0.24	0.22	0.14	0.06	0.03
Naïve Bayes	0.20	0.12	0.01	0.09	0.06

Table 5: Data mining results

The statistical analysis was performed on the experimental data using the statistics software Design Expert. A general, 4x5 fixed-effects factorial experimental design was used. Two dependent variables were tested, both the Success Rate and the Kappa Statistic. The effects model is described as follows:

$$y_{ijk} = \mu + \tau_i + \beta_j + (\tau\beta)_{ij} + \varepsilon_{ijk}, \quad \begin{cases} i = 1,2,3,4 \\ j = 1,2,3,4,5 \\ k = 1,2,...,8 \end{cases}$$

τ is the data mining algorithm factor

β is the data set factor

y is alternately the Success Rate and the Kappa Statistic

The results of the ANOVA, with the model diagnostics are presented in Appendix B. Figures 6 and 7 present the results graphically, showing the statistically significant interaction between the factors, and showing that both the Bayesian Network and the C4.5 decision tree data mining algorithms out-performed the other two methods.

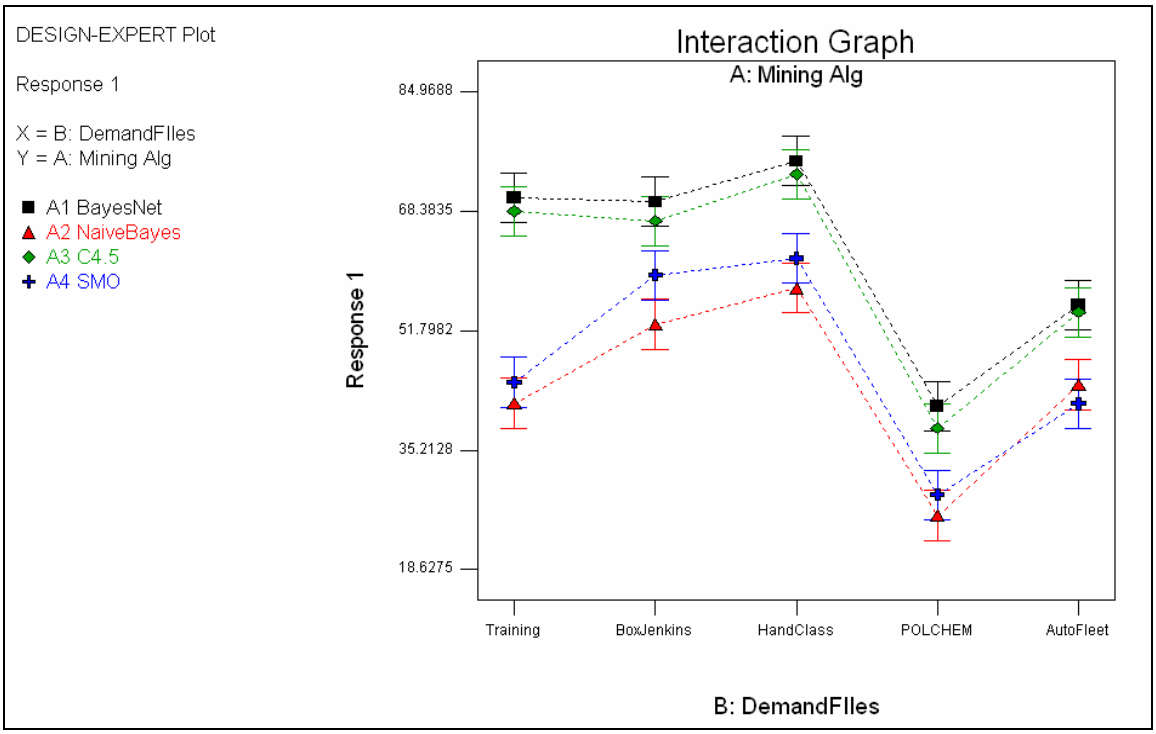


Figure 6: Success Rate results

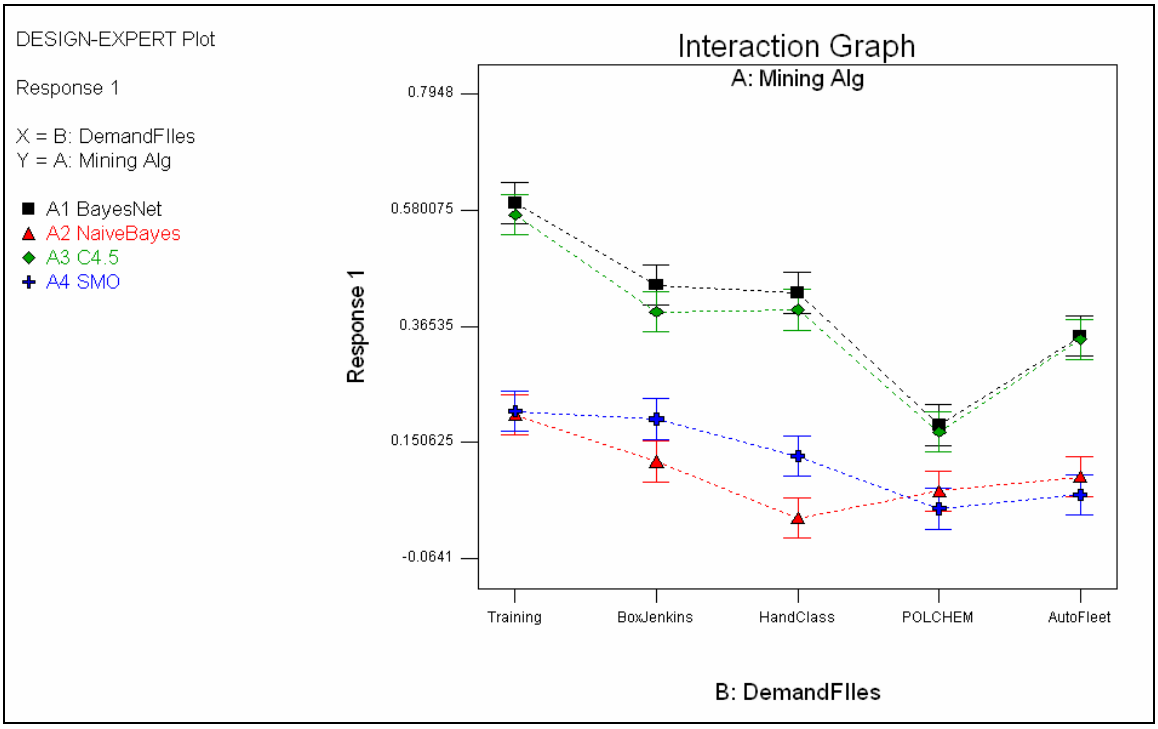


Figure 7: Kappa Statistic results

Using the archetype demand simulation to generate transaction series for use as training input to data mining algorithms appears to provide an effective means of building classification models for service part inventory items. Both the Bayesian Network and C4.5 decision tree classifiers gave good results, however the Bayesian Network's performance within WEKA is very fast, taking only seconds to classify several thousand instances, and was selected to perform the actual demand transaction classification for input to the next phase of experimentation.

Chapter 6

With the groundwork laid describing a viable method of identifying a demand pattern in the transaction stream, the next step to make this information useful, is to provide a means of applying it to the management of the inventory. To address this, the following simulation optimization experiment is presented which is path driven by bench stock inventory demand transaction streams which have been classified using the archetypal demand classification method described in the previous chapter. The objective is to determine if any of a set of common stochastic inventory management policies performs better when faced with certain types of demand patterns. The remainder of this chapter will define the inventory cost model describing the cost function being optimized, the inventory simulation will be covered and finally the method of optimization, SPSA, and its integration into the simulation will be detailed.

Inventory Cost Model

$$L(I_r, R_s, R_r, d_{R_b}, c_u, R_b, H) = 0.25I_r + 40R_s + 0.5R_r + 240(e^{\phi} - 1) + 4c_u R_b + \tfrac{0.015}{30} c_u \sum_{i=1}^{30} H_i$$

I_r is the number of inventory reviews

R_s is the number of special orders

R_r is the number of standard orders

ϕ is the back order days penalty

$\phi = \min(d_{R_b} / 7.0, 8)$ upper limit is set to 100 idle workers

$d_{R_b} \in \mathbb{R}^+$ the number of backorder days

R_b is the number of back orders

$I_r, R_s, R_r, R_b \in \mathbb{R}^+$

$I_r \leq 30$

$R_r \leq 30$

Holding cost

The holding cost estimate used in L is the sum of the cost of capital and an estimate of the variable costs such as storage, handling, shrinkage, and obsolescence. These monthly cost rates are multiplied by the average inventory stock level observed during the past simulated month (30 days).

Order placement

The order placement cost is separated into two parts, one for standard orders at a cost of $0.5 per order and the other for special expedited orders at a cost of $40 per special order. The cost constants used are estimates based upon the actual costs displayed by the operational service part inventories observed. The separation of the two cost elements is important because the normal ordering and replenishment cost involves relatively few human interventions. The special orders are assumed to require a dedicated purchasing agent or inventory manager to initiate, process, and track the order. The special order capability is modeled within the simulation and its process simulation network is selected when the order quantity for an item is found to be zero. This special order attribute is necessary to represent very slow moving items with relatively short lead times. As a management decision, certain inventory items can be set to an order quantity of zero to flag special ordering is required. An additional cost indirectly related to the ordering cost is the inventory review cost. Each time an inventory manager is required to perform a review of the inventory items a cost of $0.25 is incurred per bin.

Idle Service Worker

One of the major elements of the cost function is the idle service worker

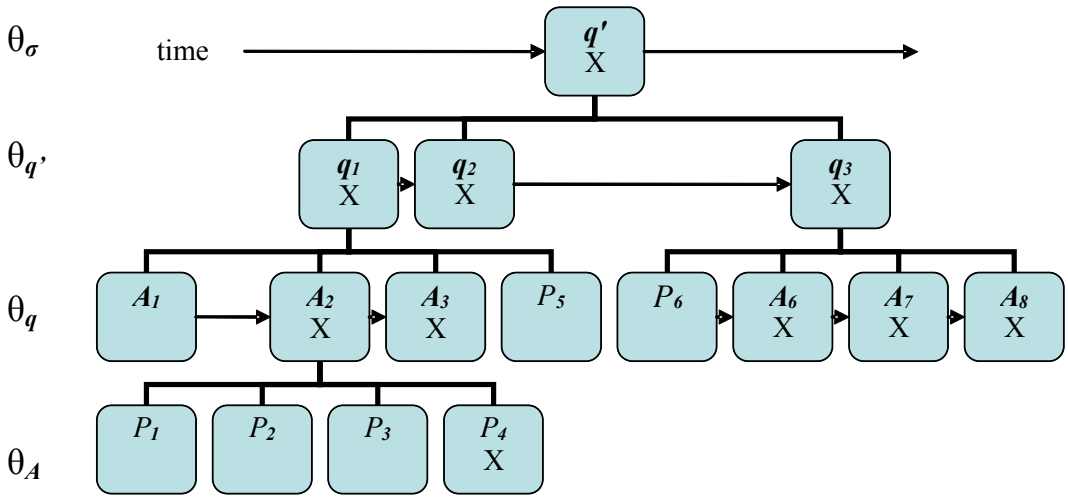


Figure 8: Stock out impact

penalty, ϕ, which is an estimate of the number of skilled worker days that will be lost due to the lack of a bench stock item. In an analysis of the independence assumption which states that the processing time of a machine is independent of past events and of the current state of the system, (Schultz et al. 1998) show that this assumption does not appear valid in low-inventory situations where processing times are affected by worker motivation. In a depot-maintenance environment where the pressure is high to complete as many maintenance operations as possible, the de-motivating influence of inventory induced delay is significant. This idle worker cost component is estimated to grow exponentially based upon the complexity of the system, the commonality or specialization of the bench stock items and the degree of coupling between the maintenance and assembly operations. The growth constant of the

exponential function can be adjusted to support these factors and also the service rate of the bench stock inventory. For the simulation, a growth constant of 1/7 is used with a maximum of total exponent of 8 which equates to approximately 100 idle workers for an entire day. Figure 8 shows a graphic depicting the impact of a stock outage on the related maintenance tasks. If item P_4 is not available, it stops the maintenance process of all the assembly operations that are dependent upon the completion of assembly A_2: A_3, q_1 , q_2 , q_3 , q', A_6 , A_7 , A_8 .

Service rate

Because these inventory items are required "raw material" for maintenance tasks and because the most valuable resources driving the maintenance process are time and skilled labor, the only acceptable service rate is 100%. The key metric for repair service is task throughput, which translates to a quick turnaround for maintenance on a complex, valuable system, which in turn increases the system's operational availability. (Sherbrooke, 2004) used operational availability throughout his analysis of optimal inventory modeling. His research focused on repairables at the component, q, level. In the U.S. Air Force these components are called LRU's, line replaceable units, because they can be replaced, as a component, on the flight line. In the analysis of several aircraft systems he found that focusing on operational availability, as opposed to hardware reliability and maintainability measures produced the best results. The key factor driving the operational availability of the repairable components described in his research is the expected number of backorders, where a backorder is defined as: $B(X \mid s) = \begin{cases} (X - s), & X > s, \\ 0 & X \le s \end{cases}$, X is a random variable

for the number of units due in and *s* is the stock level. However, this must be understood to be within the framework of a (s-1, S) inventory control policy.

The service rate modeled by the multi-item simulation used in this research is directly described with respect to the number of backorders. A backorder is defined as any request of an inventory item from warehouse bin β where the quantity requested is greater than β_H, the on hand balance in the bin. The backorder quantity is related to two factors of the cost function being minimized. The first is strictly the number of backorders weighted by an expediting, management and analysis charge of 4 times the items unit cost. The second is the idle worker penalty, ϕ, which is a function of the number of backorder days. These two cost elements act to drive the inventory positive, while the minimization of the number of regular and special orders, reviews and inventory holding cost act to drive the inventory levels negative. This pressure for a positive inventory seeks to meet that 100% service rate goal of having the right bench stock part in the right place at the right time.

Convexity

An assumption when using stochastic optimization is that the objective function be sufficiently smooth and that at least the local optimum be found at a zero gradient point. (Fu and Hill, 1997)

All of the sub functions of the cost function *L* are non-negative linear functions, $f : \mathbb{R} \to \mathbb{R}$, with exception of the exponential idle worker penalty. The exponential element of the cost function is also convex by the fact that it is twice differentiable and its Hessian is positive semi-definite. Also, any positive linear

combination of convex functions is convex. Therefore, the cost function L is convex and also differentiable.

Simulating the service part inventory

The simulation described below is the modified version of an inventory model currently in use by a Fortune 500 corporation to evaluate inventory scenarios when considering the risk or feasibility of certain business opportunities. The modifications to the model have been the addition of the multiple inventory policy enhancements. The simulation models were created using Visual SLAM and AWESIM simulation software (Pritsker and O'Reilly, 1999). The network models for these simulations are found in appendix C.

Inventory Input

The initial phase of the simulation is the input of the inventory items being modeled. The inventory file is read one record at a time and stored in an internal AWESIM data structure. If the simulation scenario which optimizes the inventory policies is selected, then after reading the inventory record, the inventory policy parameter file is read. Next, the simulation variables associated with the inventory item, which are not read from the inventory file or the parameter file, are initialized. Also, it is at this point that the initial inventory value is calculated as the product of the unit price of the inventory item and the starting inventory quantity. After these initialization steps, the inventory policy type indicated in the inventory file will cause the simulation to follow one of two possible inventory policy paradigms: an order

quantity system (r Q), or an order-up-to system (s, S). The simulation will then cycle through the chosen inventory policy until the simulation terminates.

Inventory Attribute	Description
NSN	The inventory item
STOCK	The physical on hand balance
Q	Order quantity
R	Reorder point
UPRICE	Unit price
LEADTIME	Replenishment Leadtime days
LOC	Inventory location (VMI or not)
INVPOS	The current inventory position
CD	Cost driver flag
DC	Distribution center
RECNUM	Record number
ORDERS	Number of orders in a month
REVIEWS	Number of reviews in a month
BOSTART	Start day waiting for a backorder
SPORDERS	Number of special orders in a month
BODAYS	Number of backorders in a month
REVIEWPERIOD	How often is the inventory reviewed (in days)
LASTORDERQ	The amount of the last order quantity
POLICY	Current inventory policy in use
STOCKLEVELSUM	Sum of stock level observations in a month
ORDERQTYSUM	Sum of order quantities in a month
DDEST	Estimate of daily demand mean
SDEST	Estimate of daily demand standard deviation
DEMANDTYPE	The type of demand
ERRORRATE	Review error flag

Table 6: Inventory Entity Attributes

Inventory Review Processes

The simulation has been written to model eight different inventory control policies. Four of these control policies model the reorder point system, often identified by (r, Q) referencing the reorder point "r" and the order quantity "Q". A third parameter "P" can also be manipulated to create a continuous review system (P=1) or a periodic review system (P>1). The other four policies model an "order-up-to" system commonly identified using the parameters (s, S) where "s" is the inventory level that triggers a reorder and "S" which is the target inventory level. In the (s, S)

policies, the order quantity is the difference between the inventory level at the time of the review and the desired inventory level "S". The following table summarizes the different inventory policies modeled in the simulation.

Control Policy	Reorder point	Order Quantity	Review Frequency	Description
1	r	Q	P	Periodic review, order quantity system.
2	r	Q	1	Continuous review, order quantity system.
3	Q	Q	P	Periodic review system.
4	1/2Q	Q	P	Periodic review, modified 2-bin system.
5	s	S-*IP*	P	Periodic review, order-up-to system.
6	s	S-*IP*	1	Continuous review, order-up-to system.
7	S-1	S-*IP*	P	Periodic review, base-stock system.
8	S-1	S-*IP*	1	Continuous review, base-stock system.

Table 7: Inventory Policies

Review and Replenishment

The review and replenishment processing under both the order quantity and order-up-to systems is modeled very similarly. The only differences are the method of determining the order quantity when replenishment is required and hence the quantity which updates the inventory stock level upon receipt. When determining whether an order should be placed, an estimate of the current stock level is obtained. Because of the type of inventory being modeled, one supporting maintenance activities, it is assumed a point-of-sale system is not in place to track issues from the inventory. Hence, the inventory review process is assumed to rely upon dedicated inventory management specialists examining each inventory stock location to estimate the stock level. Within the simulation, this estimate is calculated as the current actual stock level, λ, plus an error factor which is modeled as a bounded

sample, x, from a standard normal distribution $X \sim N(0,1)$:

$\hat{\lambda} = \lambda + \lambda \times \max(-0.2, \min(0.2, x))$. Therefore the resulting determination of the inventory position during the review is calculated as follows:

$IP = \hat{\lambda} + Open\ orders - Backorders$. This inventory position is then checked against the specific policy reorder point, either r or s. If $IP \le r\ or\ IP \le s$, then an order placement is simulated. The inventory position is increased by the ordered quantity and the lead time delay is calculated as a sample from a log normal distribution using the mean lead time provided from the inventory file and a standard deviation of 0.1 to introduce moderate lead time variability. (When available, the estimate of lead time standard deviation could come from the subject inventory information system through the inventory input file). After the lead time has expired, the inventory stock level λ will be increased by the ordered quantity. At this point any backorders waiting to be filled are read from the backorder file and satisfied using the current receipt quantity.

Demand Process

The demand process starts by reading the demand records from the requisition transaction file, DEMAND.TXT. Each record of the demand transaction file includes the simulation day on which the demand occurs. When the day for the demand is reached within the simulation, the demand is released from the input phase of the demand into the actual demand processing simulation model. As each demand is processed, the inventory stock level, λ, is compared with the demand quantity. If the demand quantity is less than the inventory stock level, then the demand is satisfied and processing continues on to the shipment phase. If the demand quantity is greater

than the inventory stock level, the demand processing records a backorder and places the backorder in an internal file for processing during the review and replenishment cycle.

Demand Attribute	Description
NSN	The inventory item
QTY	Demand quantity
DATE	Demand day
SHIPTIME	Time from order to dock
DELTIME	Time from dock to destination
IPG	Priority
DUPRICE	Demand unit price
STIMESAM	Modified ship time
BOFLAG	Backorder flag
DLOC	Location of inventory
DCD	Demand cost driver flag
DC1	Primary distribution center
F1	Freight charge from DC1
DC2	Secondary distribution center
F2	Freight charge from DC2
DC3	Tertiary distribution center
F3	Freight charge from DC3
DCSEL	Selected distribution center
DCF	Selected DC freight charge

Table 8: Demand Entity Attributes

Cost Variables

Some key attributes collected during the simulation provide the input to the cost function. Every inventory review, order, special order, and backorder is collected for each inventory item. These are the obvious indicators of the inventory item behavior. The number of days an inventory item has an unfilled backorder is counted to represent the number of days maintenance workers may be idle waiting for parts to continue work. The average dollar value held in inventory is also collected for each item. All of these variables and their use can be found in the statistics collection simulation network in appendix C.

Simulation Optimization

The aim of this simulation optimization experiment, as stated earlier, is to find an optimal inventory policy that matches the demand pattern that is encountered. However, the purpose is not to define an optimization method that finds the global optimum, given the cost function definition, for each of the inventory policies as applied to different demand patterns. The optimization is applied, using a viable and tested optimization method, in a greedy fashion to find a solution that is better than the starting values of the control parameters, unless we are starting at a good minimum. The experimental data for the simulation comes from actual inventory

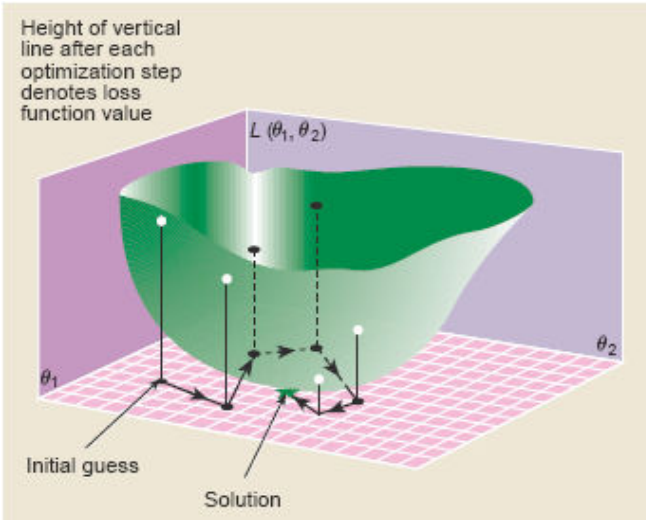


Figure 9: SPSA Execution Visualization (Spall, 1998b)

transactions and the control parameters are the parameters in use. Therefore, it can be assumed that some of them have been adjusted over time to produce results very close to optimal. The actual goal is to produce the best set of inventory parameters encountered during a limited number of iterations of the SPSA algorithm.

SPSA has been applied to the inventory problem of identifying the optimal parameters in an academic example of the (s,S) order up to policy (Fu, 2002). It has

also been used by (Spall and Cristion, 1998) to optimize the parameters controlling wastewater treatment; a model of affine-nonlinear multiplicative control form. Several other successful applications of this method are described and referenced in (Spall, 1998b), including signal timing for vehicle control, optimal targeting of weapon systems, locating buried objects using electrical conductivity, queuing systems, control of a heavy ion beam, and several others.

The general problem definition addressed is a parametric optimization problem:

$$\min_{\theta \in \Theta} J(\theta),$$

where the objective function of interest is $J(\theta) = E\left[L(\theta, \omega)\right]$. The sample of the objective function is denoted $L(\theta, \omega)$, where ω represents the stochastic effects, the sample path, of the system modeled; θ is a vector of m controllable parameters; and Θ is the constraint set on θ. The optimum of the objective is defined as follows:

$$\theta^* = \arg\min_{\theta \in \Theta} J(\theta).$$

The problem of minimizing $L(\theta, \omega)$ implies that for each trace k of the sample path ω, the solution to the following is being sought:

$$g_k(\theta_k) \equiv \frac{\partial L_k}{\partial \theta_k} = \frac{\partial u_k^T}{\partial \theta_k} \cdot \frac{\partial L_k}{\partial u_k} = 0.$$

However, the exact functioning of the system is not known, so the term $\partial L_k / \partial u_k$ is not generally computable. Therefore, $g_k(\theta_k)$ is also not explicitly available and so, the standard gradient descent optimization methods, or any algorithm relying on this term is not available. (Spall and Cristion, 1998).

Because analytical techniques relying on the gradient are not applicable in this problem, another option is to estimate the gradient using stochastic approximation and then calculating the optimal parameter set through successive iterations of the following general form of the stochastic algorithm:

$$\theta_{k+1} = \Pi_{\Theta}\left(\theta_k - a_k \hat{\nabla} J_k\right)$$

where θ_k is the parameter set at the start of trace *k*, $\hat{\nabla} J_k$ is the estimate of $g_k(\theta_k)$, a_k is a positive sequence of steps, and Π_{Θ} is a projection onto Θ. The SPSA algorithm uses the following formulation to estimate the gradient approximation of the *l*th element of the parameter vector:

$$\hat{\nabla} J_{kl} = \hat{g}_{kl}(\theta_k) = \frac{\hat{L}_k^+ - \hat{L}_k^-}{2c_k\Delta_{kl}}, l = 1,2,...,m$$

where $\hat{L}_k^+ = \hat{L}_k(\theta_k + c_k\Delta_k)$ and $\hat{L}_k^- = \hat{L}_k(\theta_k - c_k\Delta_k)$

$\{c_k\}$ is a sequence of positive numbers satisfying certain regularity conditions typically $c_k \to 0$ or $c_k = c \ \ \forall k$

$\Delta_k = (\Delta_{k1}, \Delta_{k2}, ..., \Delta_{km})^T$ is a random vector and

where $\{\Delta_{ki}\}$ are independent, bounded, symmetrically distributed about zero random variables, $\forall k, i,$ identically distributed at each *k*.

Stochastic Perturbation Analysis

The implementation of the SPSA algorithm within an optimization via simulation environment requires three sequences for proper performance and two estimates of the loss function. The first sequence $\{a_k\}$ is the step-size multiplier sequence. The second is the difference sequence $\{c_k\}$ used for the gradient estimate. These two sequences must be positive and must converge to zero at the appropriate

rate. The third sequence, $\{\Delta_k\}$ is the vector of simultaneous perturbations. According to (Spall, 1992) this sequence can effectively be generated from a series of samples from a ± 1 Bernoulli distribution which has a symmetric distribution with a mean of 0 (Fu and Hill, 1997). In addition to the sequences, the algorithm requires two estimates of the loss function $L(\cdot)$. These cost estimates are the output of the each simulation run. The three sequences are calculated using the recommendations provided in (Spall, 1998a). The difference sequence $\{c_k\}$ is calculated as $c_k = c/(k+1)^{\gamma}$, and the gain sequence $\{a_k\}$ is calculated as $a_k = a/(A+k+1)^{\alpha}$, where the values for α and γ are 0.602 and 0.101 respectively. The constant *c* is set to the minimum of the standard deviation of the ten samples of $L(\theta_0)$ and 180. The 180 maximum is established to restrict the largest perturbation increment to be half of the number of days in a year. The constant *A* which is recommended to be set to approximately 10% of the maximum number of iterations expected to find a minimum, is set to 100. The constant *a* is recommended to be calculated to achieve the desired magnitude of change applied to the parameters in the early stages of the algorithm. The perturbation sequence $\{\Delta_k\}$ is an *m*-dimensional vector of independently generated from random samples from a Bernoulli ± 1 distribution with a probability of 0.5 for each ± 1 result.

The pseudocode describing the SPSA algorithm as it was implemented for this experiment is presented below. The c-code for the implementation is provided in appendix D.

Algorithm 1: **Simultaneous Perturbation Stochastic Approximation**

spsa(<inventory file> <runs> [<control policy> <A> <gamma> <alpha>)

INVENTORY ITEMS ←**READ** inventory records
WRITE simulation inventory input file
For each INV_ITEM **in** INVENTORY ITEMS
    WRITE policy control parameter file
SIMULATE 10 runs of initial parameter setting
For all INV_ITEM **in** INVENTORY ITEMS
    READ simulation objective function output
    c[INV_ITEM**]** ← **min** (stdev of initial 10 cost results, 180)
For each INV_ITEM **in** INVENTORY ITEMS
    Ck[INV_ITEM**]** ← **c[**INV_ITEM**]**/ $(1)^{\gamma}$
SIMULATE parameters + **ck[**INV_ITEM**]**
SIMULATE parameters - **ck[**INV_ITEM**]**
For each run **in** runs
    For each INV_ITEM **in** INVENTORY ITEMS
        For each m **in** θ
            $\Delta_{run,m} \leftarrow \pm 1$ Bernoulli
        $\hat{L}^{+}_{run}$ ←**READ** simulation objective function output
        $\hat{L}^{-}_{run}$ ←**READ** simulation objective function output
        ck[INV_ITEM**]** ← **c[**INV_ITEM**]**/ $(run+1)^{\gamma}$
        For each m **in** θ
            $\hat{g}_{run,m} \leftarrow \left(\hat{L}^{+}_{run} - \hat{L}^{-}_{run}\right)/2$ **ck[**INV_ITEM**]** $\Delta_{run,m}$
            If run = 1 **then** set **a[**INV_ITEM**][**m**]**
            ak$_{run}$**[**INV_ITEM**][**m**]**← **a[**INV_ITEM**][**m**]**/ $(runs + A)^{\alpha}$
        Min← current minimum cost and parameters if seen
        MA← moving average of $\left(\hat{L}^{+}_{run} + \hat{L}^{-}_{run}\right)/2$
        If $\sum_{1}^{m} \hat{g}_{run,m}$ < 0.0001 AND runs > 10
            If current cost average < **Min then** good minimum found
        If poor local min found (**MA>Min (1.1)**) **then** backtrack to **Min**
        For each m **in** θ
            $parm_{m+1} \leftarrow parm_{m} - \mathrm{ak}_{run,m}\hat{g}_{m}$
            Enforce constraints
        WRITE policy control parameter file
        If all mins found **then continue to** 36 **else**
            SIMULATE parameters + **ck[**INV_ITEM**]**
            SIMULATE parameters - **ck[**INV_ITEM**]**
WRITE optimal parms and costs **found**

A modification to the standard SPSA algorithm was implemented for the experiment. The latest best cost average found during the execution of the algorithm is stored and used to prevent the search from ending in what is more than likely a local minimum which could be worse that the best cost minimum seen. If the 10 period moving average of $\left(\hat{L}^{+}_{run} + \hat{L}^{-}_{run}\right)\Big/2$ is greater than 110% of the best minimum seen so far, the parameters are set back to the best minimum parameters and the algorithm is allowed to continue. This backtrack paradigm is also used if the $\left(\hat{L}^{+}_{run} + \hat{L}^{-}_{run}\right)\Big/2$ exceeds double the current best min. Also no backtracking is allowed until at least 20 iterations have processed.

Simulation input data

The input data for the simulation was taken from two separate databases, one supporting aircraft maintenance and the other vehicle maintenance. The first step in the data preparation process was the classification and labeling of the transaction streams into their respective demand types using the trained Bayesian Network classifier from the experiment described in chapter 5. Once the items were labeled, extract queries were prepared to create the inventory input files and the demand transaction files. The inventory file format was the same for both the **spsa** program and the AWSIM simulation. The following table describes the input data format for both the inventory file and the demand file.

INVENTORY FILE				DEMAND FILE	
FIELD	**TYPE**	**VALUES**		**FIELD**	**TYPE**
RECNUM	INTEGER			NSN	STRING
NSN	STRING			QTY	FLOAT
STOCK	INTEGER			DATE	INTEGER
Q	FLOAT			DELTIME	FLOAT
R	FLOAT			SHIPTIME	FLOAT
REVIEWPERIOD	INTEGER			IPG	INTEGER
POLICY	INTEGER	1-8		UPRICE	FLOAT
UPRICE	FLOAT			DC1 Distribution center 1	INTEGER
LEADTIME	INTEGER			F1 Freight from DC 1	FLOAT
LOC	INTEGER	1,2		DC2 Distribution center 2	INTEGER
CD Cost Driver	INTEGER	0,1		F2 Freight from DC 2	FLOAT
DC Distribution center	INTEGER	1,2,3		DC3 Distribution center 3	INTEGER
DDEST	FLOAT			F3 Freight from DC 3	FLOAT
SDEST	FLOAT				
DEMANDTYPE	INTEGER	1,2,3,4			

Table 9: Simulation input file formats

Chapter 7

Experimentation

The experimentation presented in this chapter has been designed to determine if there is evidence that the implementation of SPSA against two separate sets of actual inventory demand data can find inventory control parameter levels that improve the overall inventory cost induced by the original settings within the inventory samples. Given that this holds, are there inventory policies that produce lower costs based upon the demand classification identified in the inventory transaction streams?

With the cost function, L, defined in the previous chapter, the SPSA optimization problem is defined as follows:

$$\min_{\theta \in \Theta} E\left[L(\cdot)\right]$$

where

$$\theta = (r, Q, P) \text{ or } (s, S, P)$$

$$\Theta = [0, r_{\max}] \times [0, Q_{\max}] \times [1, 360]$$

and where the simulation path is driven by two separate data sources. The first data source is taken from bench stock inventory transactions recorded during the operation of aircraft maintenance. A set of 1658 36-month transaction time series were extracted from the inventory management information system. These time series were then classified using the archetypal-demand-trained Bayesian Network classifier. From this set of classified demand traces, 45 items of each type of demand were selected at random. The inventory and demand data for these 45 items was assembled and recorded in the files described above for input into the SPSA and

simulation programs. The second data source was taken from an inventory management system controlling a large military automotive vehicle fleet. The same process was used to extract the data and a random sample of 10 items of each demand type was selected from a total of 1298 inventory transaction streams. The number of inventory items and the demand transactions in each set of data is provided in the table below:

Data Source	Inventory Items	Periodic Demands	Seasonal Demands	Level Demands	Sparse Demands
Aircraft	45	1026	755	818	178
Automotive	10	4776	14496	29565	1434

Table 10: Test data metrics

All of the tests were run on the same workstation running a production version of AWESIM 3.0. The processor was an Intel® Pentium® 4 running at a clock speed of 2.80 GHz, executing with 1GB of RAM. The operating system software was Microsoft Windows XP Professional version 2002, service pack 2.

The statistical analysis was performed on the experimental data using the statistics software Design Expert. A general, 2x4x8 fixed-effects factorial experimental design was used. Three dependent variables were tested, optimization cost mean delta, the inventory cost mean and the inventory cost standard deviation over the 10 optimized-parameter simulation runs. The effects model is described as follows:

$$y_{ijkl} = \mu + \tau_i + \beta_j + \gamma_k + (\tau\beta)_{ij} + (\tau\gamma)_{ik} + (\beta\gamma)_{jk} + (\tau\beta\gamma)_{ijk} + \varepsilon_{ijkl}, \quad \begin{cases} i = 1,2 \\ j = 1,2,3,4 \\ k = 1,2,\ldots,8 \\ l = 1,2,\ldots,n \end{cases}$$

τ is the review error factor
β is the demand pattern factor
γ is the inventory policy factor
y is alternately the Optimization Cost Delta, the Inventory Cost Mean, and Inventory Cost Standard Deviation

Each of the datasets was run through the SPSA simulation optimization for a maximum of 40 iterations, fewer executions if all the minimums in a policy-demand dataset were found. 2880 total SPSA runs were conducted for the aircraft data and 640 runs for the vehicle data. During the first phase of the simulation optimization, using the inventory control parameters for each item at a starting point θ_0, the simulation is executed for 10 iterations and the average cost resulting from this run is stored as the initial cost. The SPSA algorithm is then executed and after it has selected a set of optimized parameters, the simulation is run using these parameters again for 10 iterations. This final cost value is the *inventory cost mean* dependent variable, the standard deviation of the cost is the *inventory cost standard deviation* dependent variable and the difference between the initial cost mean and the final cost mean is the *optimization cost delta* dependent variable.

In addition, it is assumed the aircraft maintenance facility is a large aircraft repair depot operation employing a large workforce. Therefore, the cost parameters for the idle worker are set, as described in chapter 6, to model the cost of up to 100 idle workers. The vehicle repair service is tested under two maintenance facility workforce levels: large and small. The large workforce idle worker cost is set identical to the large aircraft maintenance facility, whereas the small workforce idle worker cost is set to model the cost of up to 12 idle workers.

The full results of the ANOVA, with the model diagnostics are presented in Appendix E. Summary data, without logarithmic transformation, is presented in Appendix F. Figures 12-24 present the results graphically and the ANOVA summary tables are presented in tables 11-19.

Aircraft depot maintenance part data

Source	Sum of Squares	DF	Mean Square	F Value	Prob > F	
Model	1372.96	63	21.79	3.12	< 0.0001	significant
Review A	0.44	1	0.44	0.063	0.8020	
Demand B	315.70	3	105.23	15.06	< 0.0001	significant
Policy C	399.28	7	57.04	8.16	< 0.0001	significant
AB	60.49	3	20.16	2.88	0.0345	significant
AC	129.20	7	18.46	2.64	0.0101	significant
BC	288.82	21	13.75	1.97	0.0054	significant
ABC	179.03	21	8.53	1.22	0.2228	
Pure Error	19682.82	2816	6.99			
Cor Total	21055.78	2879				

Table 11: ANOVA for aircraft inventory cost mean

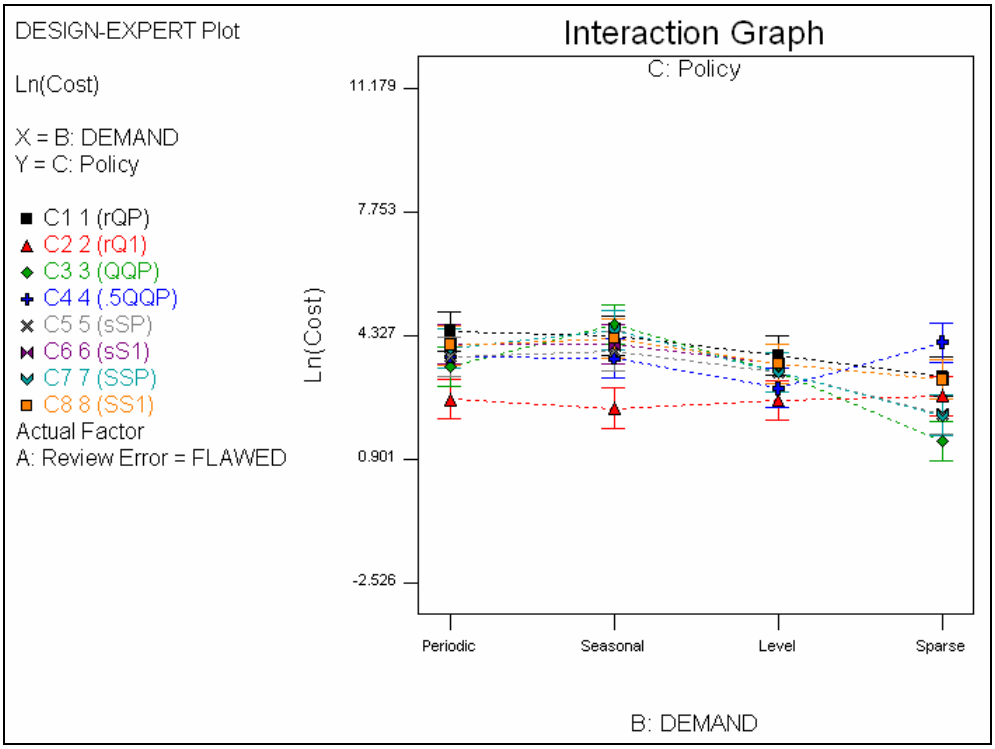


Figure 10: Aircraft inventory cost mean, flawed reviews

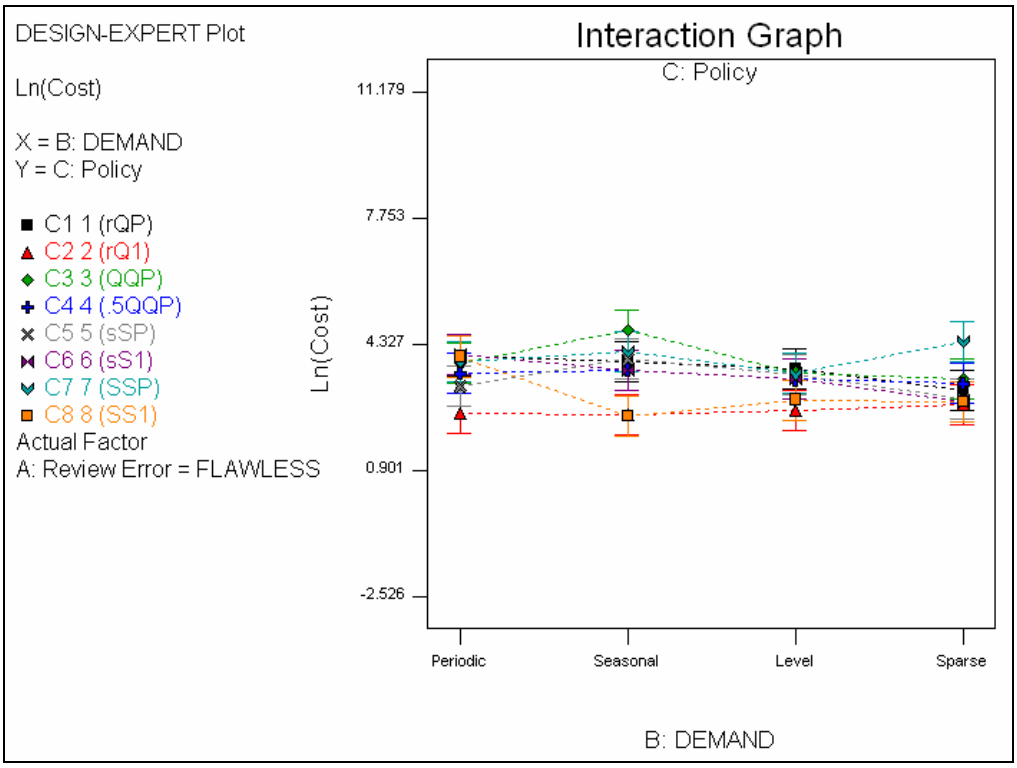


Figure 11: Aircraft inventory cost mean, flawless reviews

Source	Sum of Squares	DF	Mean Square	F Value	Prob > F	
Model	1.809E+010	63	2.872E+008	3.76	< 0.0001	significant
Review A	7.848E+007	1	7.848E+007	1.03	0.3110	
Demand B	1.855E+008	3	6.185E+007	0.81	0.4887	
Policy C	1.356E+010	7	1.937E+009	25.35	< 0.0001	significant
AB	3.107E+008	3	1.036E+008	1.35	0.2548	
AC	9.019E+008	7	1.288E+008	1.69	0.1079	
BC	2.135E+009	21	1.017E+008	1.33	0.1434	
ABC	9.202E+008	21	4.382E+007	0.57	0.9382	
Pure Error	2.152E+011	2816	7.644E+007			
Cor Total	2.333E+011	2879				

Table 12: ANOVA for aircraft optimization delta

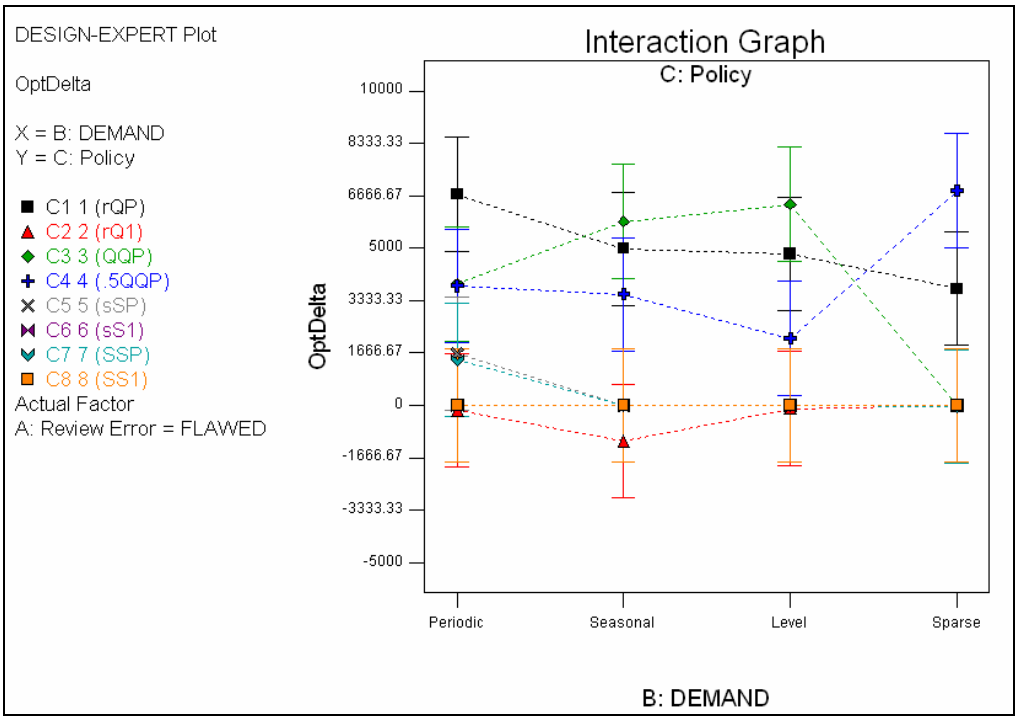


Figure 12: Aircraft optimization delta, flawed reviews

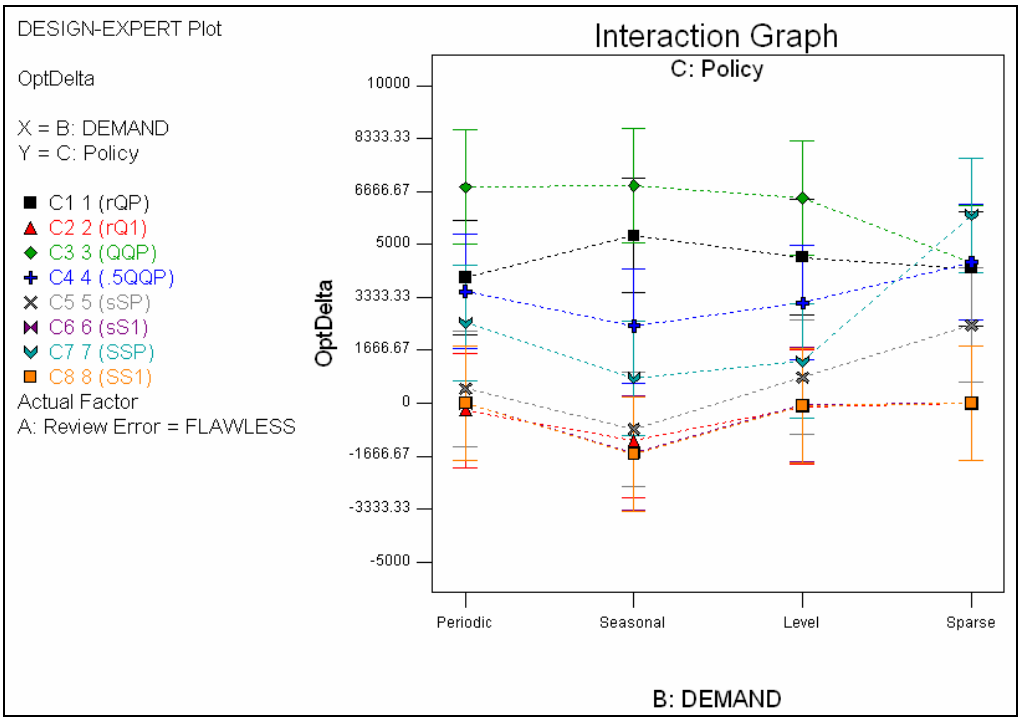


Figure 13: Aircraft optimization delta, flawless reviews

Source	Sum of Squares	DF	Mean Square	F Value	Prob > F	
Model	1.109E-013	63	1.761E-015	8.32	< 0.0001	significant
Review A	2.969E-016	1	2.969E-016	1.40	*0.2363*	
Demand B	1.166E-015	3	3.885E-016	1.84	*0.1385*	
Policy C	8.664E-014	7	1.238E-014	58.48	*< 0.0001*	significant
AB	2.976E-015	3	9.921E-016	4.69	*0.0029*	significant
AC	2.128E-015	7	3.040E-016	1.44	*0.1860*	
BC	9.669E-015	21	4.604E-016	2.18	*0.0015*	significant
ABC	8.064E-015	21	3.840E-016	1.81	*0.0130*	significant
Pure Error	5.960E-013	2816	2.116E-016			
Cor Total	7.069E-013	2879				

Table 13: ANOVA for aircraft cost standard deviation

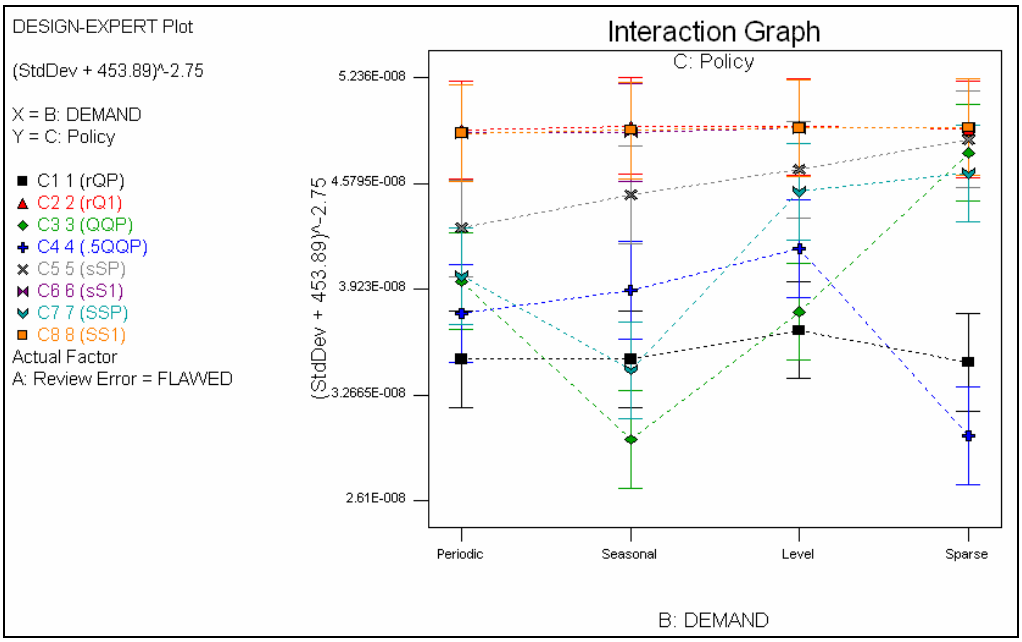


Figure 14: Aircraft cost standard deviation (transformed), flawed reviews

Military vehicle maintenance data, large workforce scenario

Source	Sum of Squares	DF	Mean Square	F Value	Prob > F	
Model	530.03	63	8.41	1.31	0.0613	not significant
Review A	4.87	*1*	4.87	0.76	0.3840	
Demand B	176.92	*3*	58.97	9.19	< 0.0001	significant
Policy C	234.80	*7*	33.54	5.23	< 0.0001	significant
AB	3.95	*3*	1.32	0.21	0.8929	
AC	16.57	*7*	2.37	0.37	0.9203	
BC	58.35	*21*	2.78	0.43	0.9878	
ABC	34.57	*21*	1.65	0.26	0.9997	
Pure Error	3696.64	576	6.42			
Cor Total	4226.67	639				

Table 14: ANOVA results for large workforce vehicle inventory cost mean

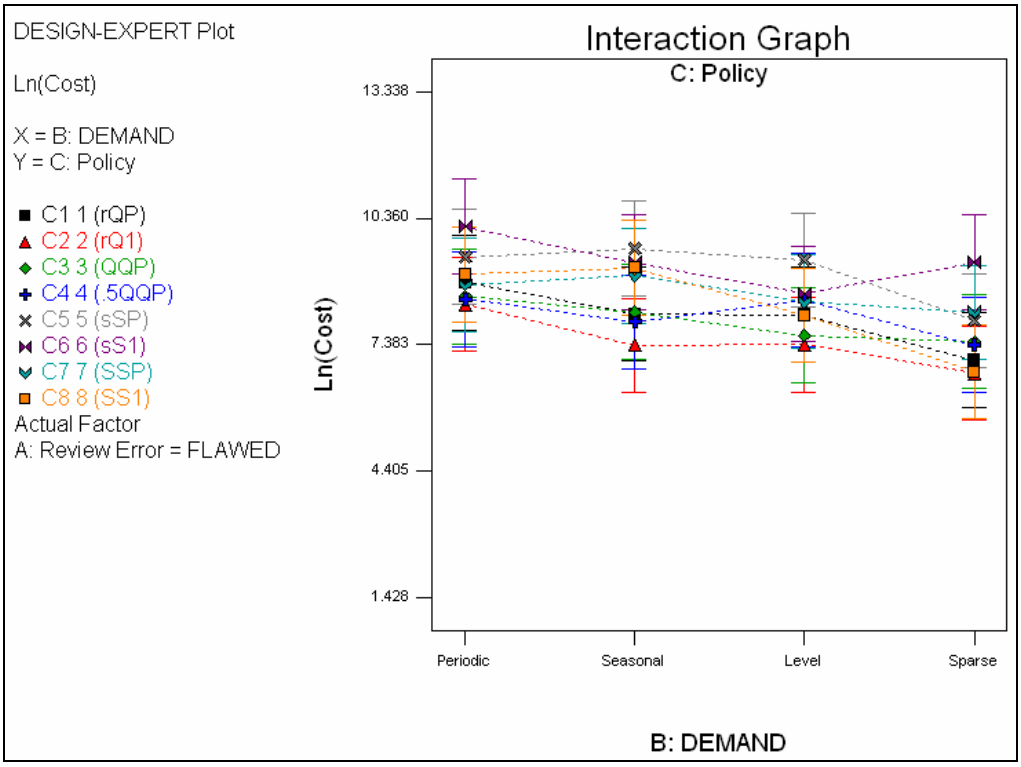


Figure 15: Large workforce vehicle inventory cost, flawed reviews

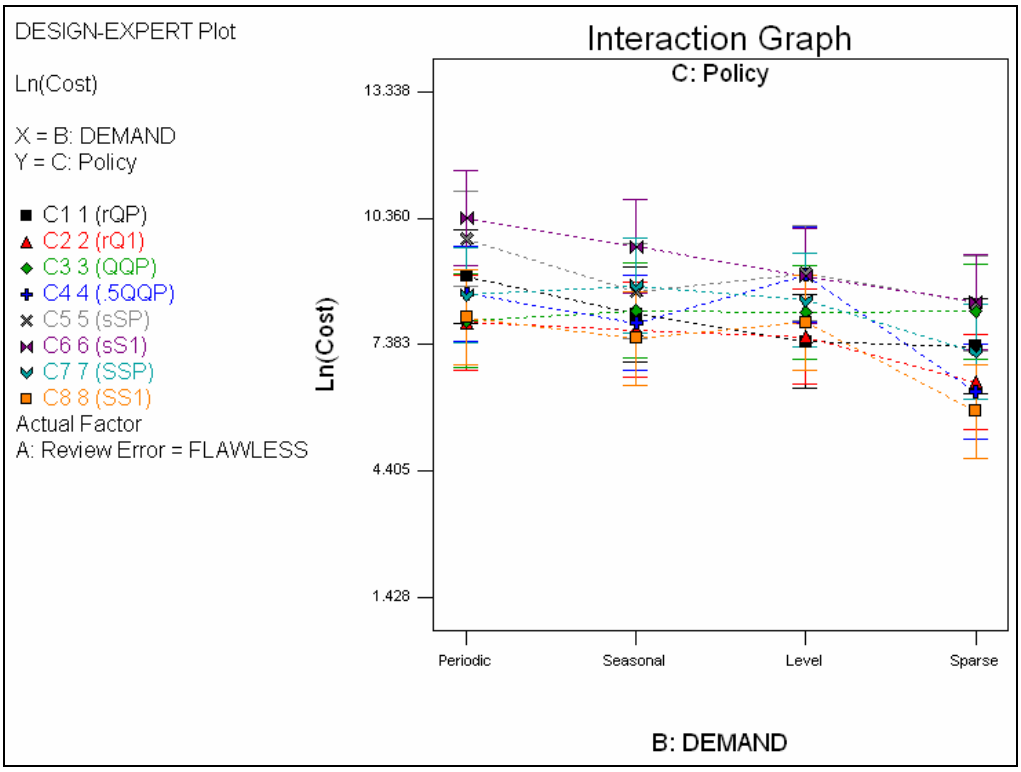


Figure 16: Large workforce vehicle inventory cost, flawless reviews

Source	Sum of Squares	DF	Mean Square	F Value	Prob > F	
Model	4.516E+012	63	7.167E+010	6.55	< 0.0001	significant
Review A	8.718E+009	1	8.718E+009	0.80	0.3726	
Demand B	1.266E+012	3	4.220E+011	38.54	< 0.0001	significant
Policy C	2.583E+012	7	3.689E+011	33.69	< 0.0001	significant
AB	5.270E+009	3	1.757E+009	0.16	0.9229	
AC	1.479E+010	7	2.113E+009	0.19	0.9869	
BC	6.195E+011	21	2.950E+010	2.69	< 0.0001	significant
ABC	1.872E+010	21	8.916E+008	0.081	1.0000	
Pure Error	6.307E+012	576	1.095E+010			
Cor Total	1.082E+013	639				

Table 15: ANOVA results for large workforce vehicle optimization delta

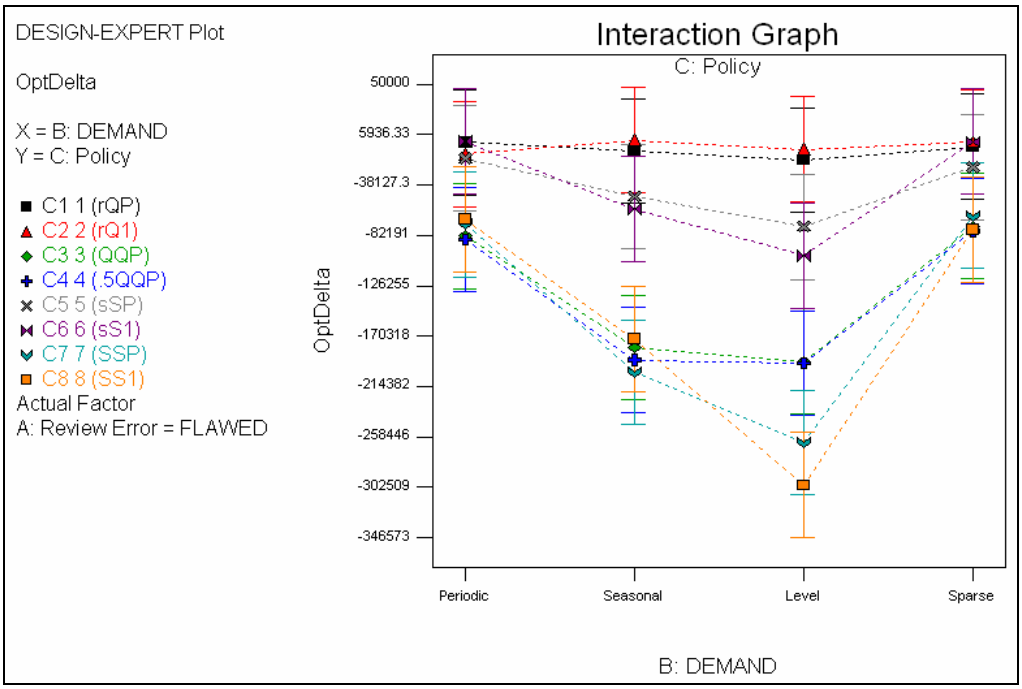


Figure 17: Large workforce vehicle optimization delta, flawed reviews

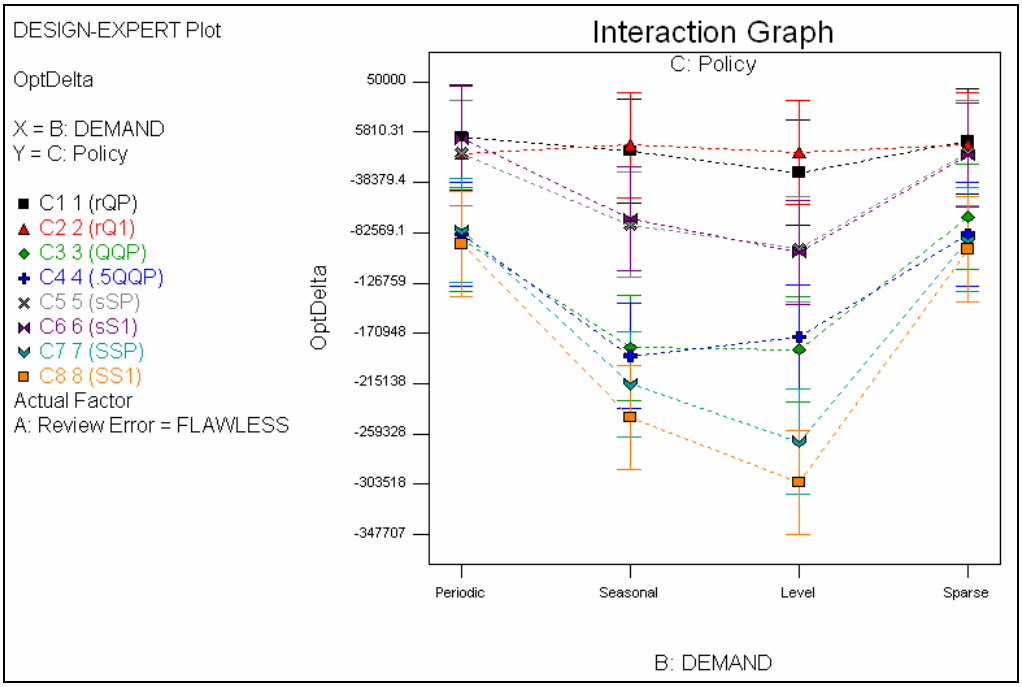


Figure 18: Large workforce vehicle optimization delta, flawless reviews

Source	Sum of Squares	DF	Mean Square	F Value	Prob > F	
Model	0.016	42	3.912E-004	2.08	0.0001	significant
Review A	5.062E-004	1	5.062E-004	2.69	0.1014	
Demand B	6.588E-003	3	2.196E-003	11.67	< 0.0001	significant
Policy C	4.927E-003	7	7.039E-004	3.74	0.0006	significant
AB	3.153E-004	3	1.051E-004	0.56	0.6425	
AC	3.843E-004	7	5.490E-005	0.29	0.9571	
BC	3.708E-003	21	1.766E-004	0.94	0.5403	
Residual	0.11	597	1.881E-004			
Lack of Fit	1.094E-003	21	5.209E-005	0.27	0.9996	
Pure Error	0.11	576	1.931E-004			
Cor Total	0.13	639				

Table 16: ANOVA for large workforce vehicle cost standard deviation

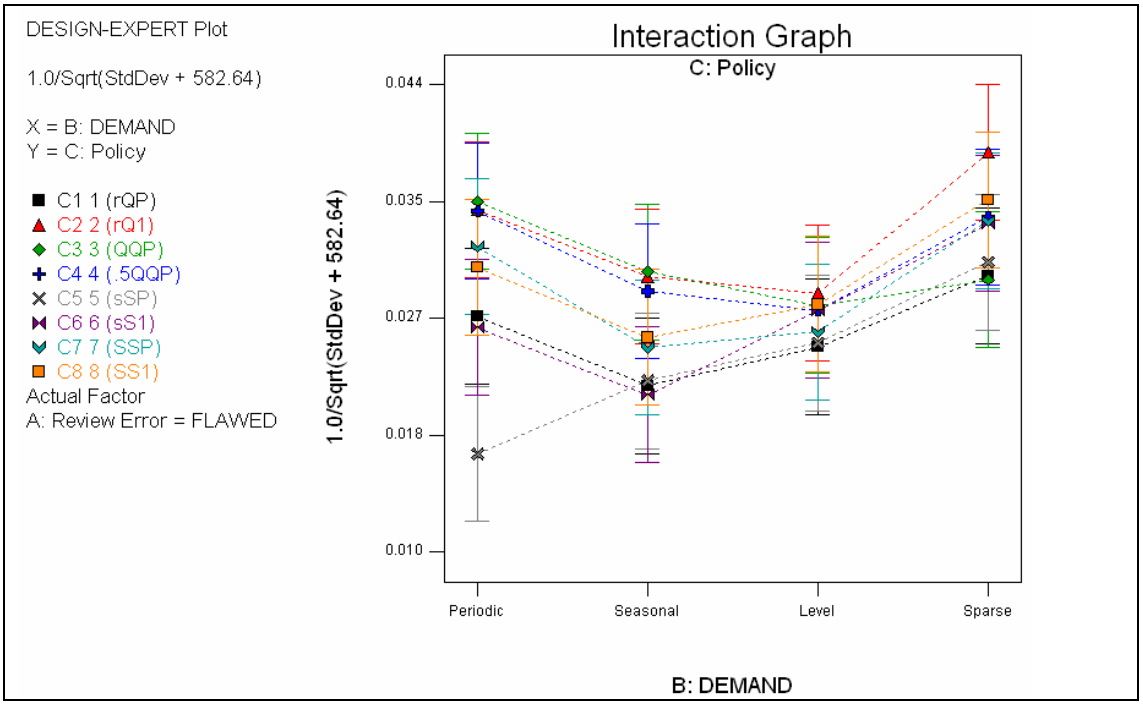


Figure 19: Large workforce vehicle cost standard deviation (transformed), flawed reviews

Military vehicle maintenance data, small workforce scenario

Source	Sum of Squares	DF	Mean Square	F Value	Prob > F	
Model	476.14	42	11.34	2.63	< 0.0001	significant
Review A	1.68	1	1.68	0.39	0.5324	
Demand B	266.08	3	88.69	20.57	< 0.0001	significant
Policy C	132.44	7	18.92	4.39	< 0.0001	significant
AB	6.90	3	2.30	0.53	0.6596	
AC	14.97	7	2.14	0.50	0.8379	
BC	54.08	21	2.58	0.60	0.9218	
Residual	2574.58	597	4.31			
Lack of Fit	12.16	21	0.58	0.13	1.0000	
Pure Error	2562.42	576	4.45			
Cor Total	3050.72	639				

Table 17: ANOVA results for small workforce vehicle inventory cost mean

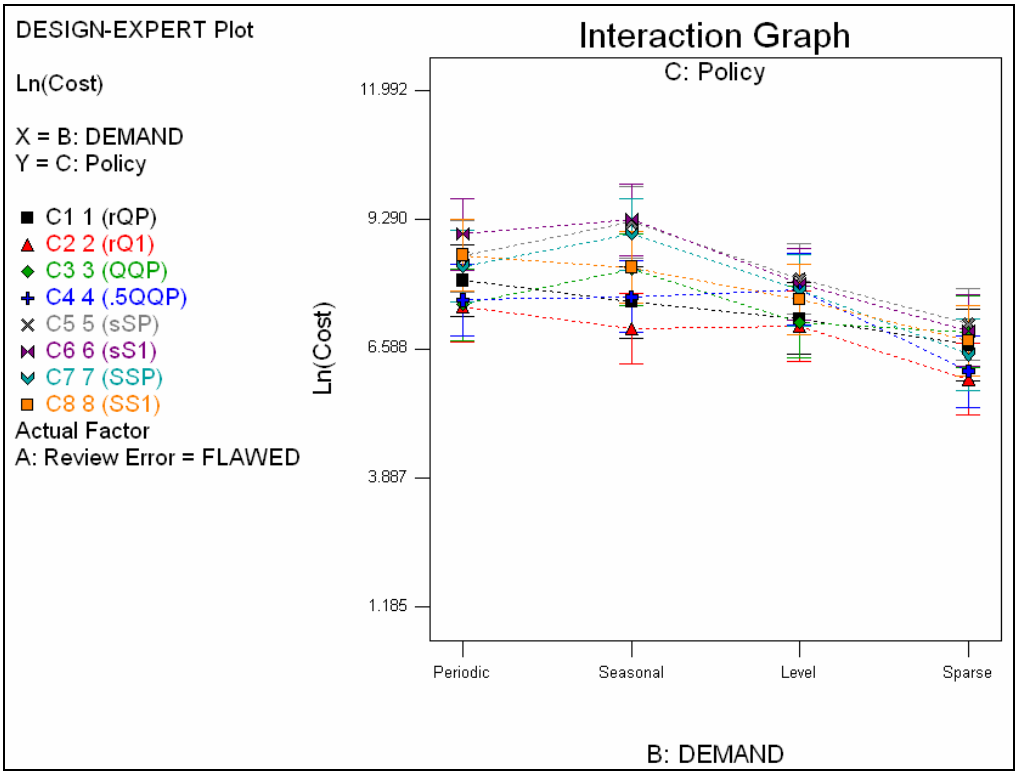


Figure 20: Small workforce vehicle inventory cost mean, flawed reviews

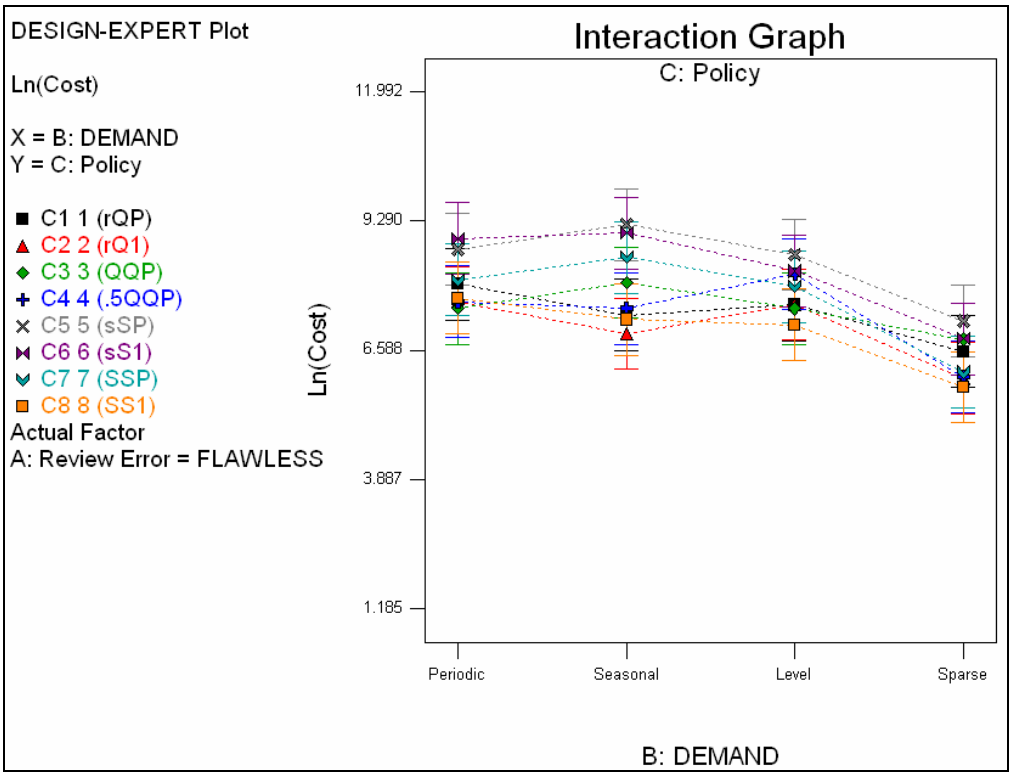


Figure 21: Small workforce vehicle inventory cost mean, flawless reviews

Source	Sum of Squares	DF	Mean Square	F Value	Prob > F	
Model	2.621E+011	42	6.241E+009	7.03	< 0.0001	significant
Review A	9.917E+008	1	9.917E+008	1.12	0.2910	
Demand B	8.349E+010	3	2.783E+010	31.35	< 0.0001	significant
Policy C	1.260E+011	7	1.800E+010	20.28	< 0.0001	significant
AB	7.689E+008	3	2.563E+008	0.29	0.8336	
AC	7.564E+008	7	1.081E+008	0.12	0.9968	
BC	5.012E+010	21	2.386E+009	2.69	< 0.0001	significant
Residual	5.300E+011	597	8.877E+008			
Lack of Fit	1.331E+009	21	6.338E+007	0.069	1.0000	
Pure Error	5.286E+011	576	9.178E+008			
Cor Total	7.921E+011	639				

Table 18: ANOVA results for small workforce vehicle optimization delta

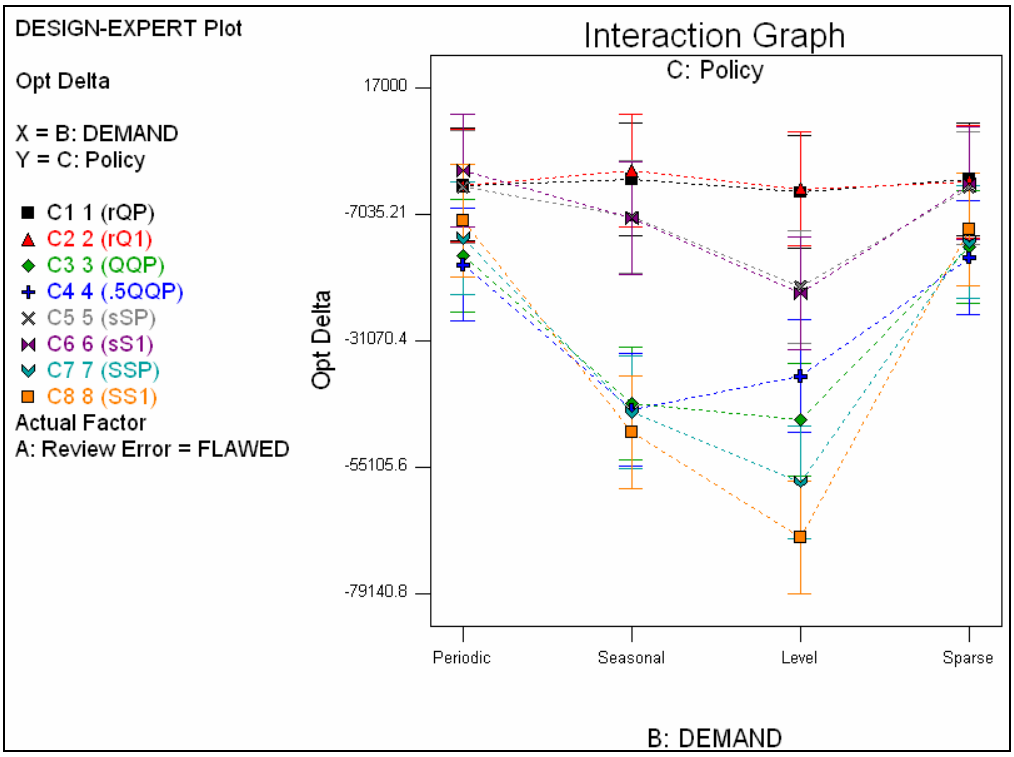


Figure 22: Small workforce vehicle optimization delta, flawed reviews

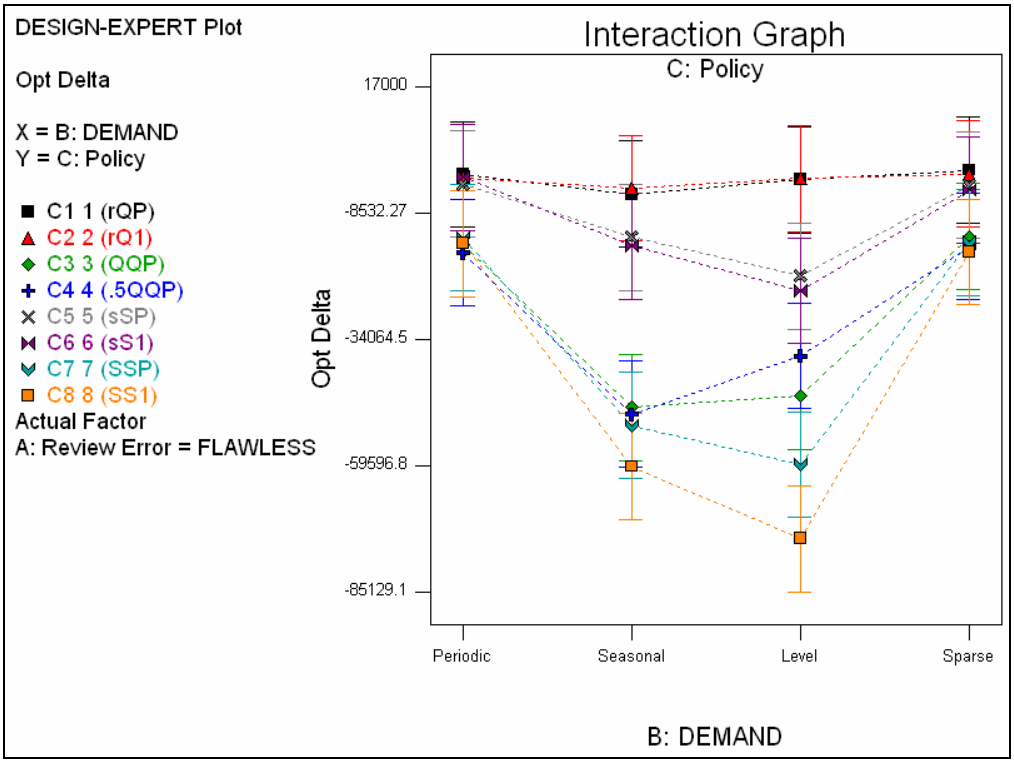


Figure 23: Small workforce vehicle optimization delta, flawless reviews

Source	Sum of Squares	DF	Mean Square	F Value	Prob > F	
Model	4.220E-004	42	1.005E-005	3.10	< 0.0001	significant
Review A	1.627E-005	1	1.627E-005	5.02	0.0255	
Demand B	1.563E-004	3	5.210E-005	16.06	< 0.0001	significant
Policy C	1.560E-004	7	2.228E-005	6.87	< 0.0001	significant
AB	1.732E-005	3	5.773E-006	1.78	0.1498	
AC	1.244E-005	7	1.777E-006	0.55	0.7982	
BC	6.366E-005	21	3.031E-006	0.93	0.5456	significant
Residual	1.936E-003	597	3.243E-006			
Lack of Fit	1.921E-005	21	9.149E-007	0.27	0.9995	
Pure Error	1.917E-003	576	3.328E-006			
Cor Total	2.358E-003	639				

Table 19: ANOVA results for small workforce vehicle cost standard deviation

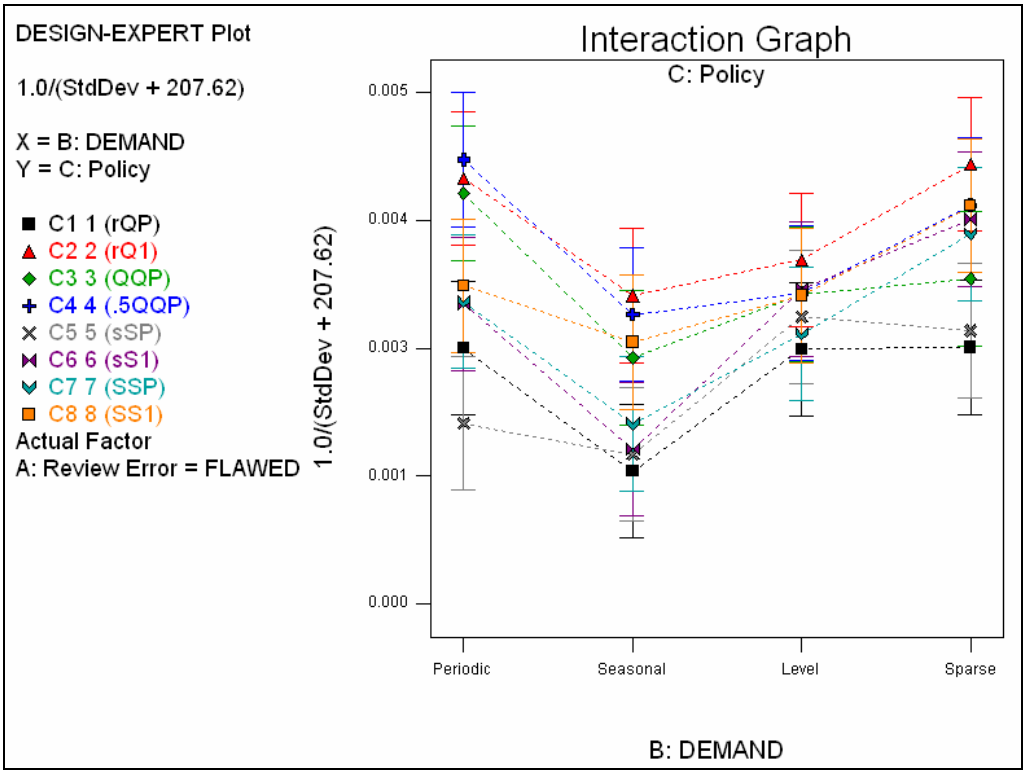


Figure 24: Small workforce vehicle inventory cost standard deviation, flawed reviews

Results

The two different data sets produced what appear to be very different results, yet, upon closer examination, a set of inventory policies do appear that perform better than others in certain demand circumstances. However, first addressing whether the

simulation optimization reduced the inventory costs of the respective inventory samples, the SPSA optimization performed quite differently for the two data sets. The aircraft bench stock inventory did not appear to realize significant cost improvement due to the optimization step. In fact, it appears just the opposite in the case of three of the inventory policies, and the other five policies made very little or no impact on improving the inventory cost level. The vehicle data however, produced significant improvement in the inventory cost position. Four of the policies showed significant cost improvement. It is also important when examining the results to do so in light of the current policies in use: (.5Q,Q,P) for the aircraft inventory and (r,Q,1) for the vehicle inventory.

By combining the results of the three dependent variables, the inventory cost mean, the cost standard deviation and the optimization delta (looking at the flawed review charts above), a reasonable policy-to-demand pattern pairing can be derived. The first step in determining the demand-to-policy pairings is to select all of the policies within a demand category whose inventory cost means fall within the 95% confidence interval of the policy with the lowest inventory cost mean. Next, of these low-cost policies, exclude all the policies that showed a positive optimization delta; in other words on average the optimization of the policy resulted in a higher cost inventory posture than the initial policy parameters. Finally, select the policy that produces the lowest inventory cost standard deviation: this is the selected policy for the given demand category. The results of this process are provided in table 20 below. The candidate policies shown produced the lowest inventory cost mean using the optimized control parameters. The values for the inventory cost standard

deviations and the optimization deltas are reported and the selected policies are in bold.

Data set	Demand	Candidate Policy	Cost Mean	Cost Standard Deviation	Optimization Delta
Aircraft Large workforce	Periodic	**(r,Q,1)**	**2.54307**	**4.91023E-8**	**-173.011**
	Seasonal	**(r,Q,1)**	**2.30602**	**4.93506E-8**	**-1134.01**
	Level	**(r,Q,1)**	**2.51491**	**4.93053E-8**	**-105.95**
		(.5Q,Q,P)	2.85412	4.17394E-8	2129.5
	Sparse	**(Q,Q,P)**	**1.3934**	**4.7689E-8**	**-0.449778**
Vehicle Large workforce	Periodic	**(r,Q,P)**	**8.84583**	**0.0271314**	**-927.145**
		(r,Q,1)	8.33211	0.034888	-11035
		(Q,Q,P)	8.52308	0.035496	-82666.8
		(.5Q,Q,P)	8.45752	0.034782	-85910
		(S,S,P)	8.7893	0.0322231	-72416.2
		(S,S,1)	9.03956	0.0306862	-68220.8
	Seasonal	**(r,Q,P)**	**8.10373**	**0.0220831**	**-8400.23**
		(r,Q,1)	7.36809	0.0300454	1166.05
		(Q,Q,P)	8.15165	0.0303956	-180252
		(.5Q,Q,P)	7.91557	0.289754	-191206
	Level	**(r,Q,P)**	**8.09291**	**0.0249112**	**-16459.8**
		(r,Q,1)	7.38253	0.0288339	-6780.68
		(Q,Q,P)	7.60612	0.0279414	-192637
		(.5Q,Q,P)	8.42958	0.0275927	-193928
		(S,S,P)	8.39842	0.0260118	-263474
		(S,S,1)	8.07551	0.0280055	-300616
	Sparse	(r,Q,P)	7.01263	0.030106	-4794.24
		(r,Q,1)	6.71115	0.0390504	-542.471
		(Q,Q,P)	**7.45851**	**0.0298213**	**-74218.5**
		(.5Q,Q,P)	7.37347	0.0343682	-78310.5
		(S,S,1)	6.74031	0.0356299	-77125.5
Vehicle Small workforce	Periodic	**(r,Q,P)**	**8.01664**	**0.00251985**	**-89.889**
		(r,Q,1)	7.47509	0.00416142	-2077.22
		(Q,Q,P)	7.51216	0.0040159	-14550.3
		(.5Q,Q,P)	7.61175	0.00434166	-16435.3
	Seasonal	**(r,Q,P)**	**7.56874**	**0.00133169**	**-1358.86**
		(r,Q,1)	7.0111	0.00302671	2.78E-010
		(.5Q,Q,P)	7.67632	0.00284434	-42060.4
	Level	**(r,Q,P)**	**7.22146**	**0.00250973**	**-2470.58**
		(r,Q,1)	7.06167	0.00337218	-2038.91
		(Q,Q,P)	7.14647	0.00304297	-46482.9
		(S,S,1)	7.62061	0.00302644	-71836.3
	Sparse	(r,Q,P)	6.67353	0.00252734	728.21
		(r,Q,1)	5.95515	0.00429523	-336.703
		(.5Q,Q,P)	6.11937	0.00390234	-15590.6
		(S,S,P)	**6.46836**	**0.00362277**	**-12105.8**

Table 20: Selecting the policy-to-demand pairing

Table 21 displays a summary of the policy-to-demand pairing. The bold **X** in a cell indicates that the pairing is the primary selection, a small x indicates alternates. As could be expected, the dynamics of the very different transaction volumes between the two data sets may explain some of the differences. However, the differences may also be ultimately tied to the different demand process dynamics that govern each of the inventory's behavior.

Data set	Policy	Periodic	Seasonal	Level	Sparse
Aircraft Large workforce	(r,Q,P)				
	(r,Q,1)	**X**	**X**	**X**	
	(Q,Q,P)				**X**
	(.5Q,Q,P)				
	(s,S,P)				
	(s,S,1)				
	(S,S,P)				
	(S,S,1)				
Vehicle Large workforce	(r,Q,P)	**X**	**X**		**X**
	(r,Q,1)			**X**	
	(Q,Q,P)			x	
	(.5Q,Q,P)				
	(s,S,P)				
	(s,S,1)				
	(S,S,P)				
	(S,S,1)	x			x
Vehicle Small workforce	(r,Q,P)	**X**	**X**	**X**	
	(r,Q,1)				
	(Q,Q,P)				
	(.5Q,Q,P)				x
	(s,S,P)				
	(s,S,1)				
	(S,S,P)				**X**
	(S,S,1)				

Table 21: Policy-to-demand pattern pairings

It is also interesting to notice the significant impact that the flawed inventory review environment had upon the performance of some of the inventory policies. Some policies appear to be more robust than others in the presence of the review errors. Notice the performance of the (r,Q,1) policy in both the aircraft and the

vehicle charts. It appears to behave the same in both the flawed and flawless inventory review environments.

Chapter 8

Conclusions

Several key points have been described and expanded upon within the research presented here which address new and significant contributions to the existing inventory management literature corpus. First, a simulation of repair service demand processes effectively created demand transaction time series representing archetypal demand structures. These time series were successfully used to train data mining algorithms which were then used to classify unseen repair service inventory transactions into categories represented by each of the archetypal demand structures. Second, through the use of SPSA simulation optimization, a means of identifying inventory policies which perform best within a set of stochastic inventory policies in the presence of certain archetypal demand was demonstrated with significant results. Third, it was shown that the applied results of this research provide an effective method of pairing classified archetypal demand with an efficiently performing inventory policy. Fourth, the inventory review process modeled within the multi-item inventory simulation showed that flawed knowledge of the actual stock level for an inventory item can significantly affect the performance of stochastic inventory policies. However, it also showed that some inventory policies appear more robust in the presence of this lack of accurate control information.

Both the data mining of the inventory time series and the performance of the SPSA optimization were not ideal. However, this was not a condition for their use, and showing that each of these methods is flawless was not an objective. Using data mining of the inventory transactions, in both the search for a leading indicator of

stock outs and the identification of demand patterns, was shown to produce acceptable results within the complex environment described. The application of SPSA simulation optimization to this type of inventory transaction data is new, and, used as a tool for detecting performance of an inventory policy, it performed adequately. In fact, recalling the two fundamental principles of modern operations management under uncertainty;

- point forecasts are meaningless and should be replaced by range forecasts
- aggregate forecasts are more accurate than individual forecasts

seeking focused single point answers in a stochastic environment does not lead to robust solutions.

Finally, the underlying question to be answered: is this a viable method for providing guidance for large-scale bench stock or repair service inventory control? The answer is yes, with caveat. The pre-requisites for applying the method require an inventory management information system which is capturing the demand information and stores enough of a demand history to drive a simulation. A simulation model of the inventory demand and control processes is required to generate the archetypal demand and for use within the SPSA simulation optimization. An objective function which reasonably captures the costs of the inventory must be defined, convex and sufficiently smooth. And, of course, the software and systems to perform the simulation and the data mining must be available and properly configured. Given these pre-requisites are in place, the research presented here shows that a method of defining a set of inventory control guidelines could be produced to

effectively assist an inventory manager in their work of optimizing repair service inventory performance.

Future Research

During the course of this research other areas of exploration appeared as promising avenues of discovery and analysis. The repair service part inventory is not unlike other consumable inventories such as green grocery and partially prepared restaurant foods, and medical supplies. Evaluating the applicability of the methods described here to inventories of other domains could produce inventory management procedures that yield beneficial cost and service results.

Related to researching other inventory domains, other categories of demand could be modeled through simulation or other means for use as input training files for data mining algorithms. The demand characterization data mining presented here examines one set of demand characteristics that are important to repair service inventory management. Additional methods of exploiting the demand characterization could also provide an avenue for beneficial research. A connection could possibly be found between a class of demand and the performance of a particular traditional demand forecasting method.

The demand attribute used in this research contained only information about the number of items demanded. Examining the impact of multi-attribute demand within a time series could provide a fruitful avenue of exploration. The exploitation of demand monetary-value time series data mining could produce results that improve value stream management and optimization.

Multiple cost function modeling within a single inventory simulation could produce results that improve the management of a complex, multi-item inventory. If information related to the criticality or the commonality of an inventory item were available, perhaps cost models could be defined which use this information to more accurately reflect the cost impact of a class of inventory items. With the ability to switch between the different cost models, perhaps a more representative cost profile could be evaluated through the use of SPSA simulation optimization.

Given a full implementation of the methods described here, additional research could focus on the heuristics that guide the selection of the inventory control policy. With the output from the demand classification and the SPSA simulation optimization, a set of heuristics could be developed that leverage this output as input to an automatic policy selection module for use by the inventory manager.

The inventory model used within this research presented a multi-item single echelon repair service inventory. Research could be directed toward application of the methods described here to the multi-echelon inventory environment. The demand structures at the different echelons could be used to guide the policy selection at each level as described above, and perhaps the simulation optimization of the inventory could focus on system wide inventory cost and service optimization.

References

1. Akcali, E., Hamlin, R. D., 1997, Teyner, T., Uzsoy, R., and Venkatachalam, G., 1997, "Spare parts inventory management for semiconductor wafer fabrication facilities", *Proceedings of the Sixth International Symposium on Semiconductor Manufacturing*, San Francisco, CA, October 1997.
2. Angerhofer, B. J., and Angelides, M. C., 2000, "System dynamics modelling in supply chain management: research review", *Proceedings of the 32nd Winter Simulation Conference*, December 10-13, 2000, Orlando, FL.
3. Beardslee, E.A. and Trafalis, T.B., 2005, "Data mining methods in a metrics-deprived inventory transactions environment", *Data Mining VI: Data Mining, Text Mining and their Business Applications,* edited by A. Zanasi, C.A. Brebbia and N.F.F. Ebecken, pp. 513-522, (WITPress, Southhampton), 2005.
4. Berman, O., Kaplan, E. H., and Shimshak, D. G., 1993, "Deterministic approximations for inventory management at service facilities", *IIE Transactions*, 1993, **25**(5), pp. 98-104.
5. Bertsimas, D., and Thiele, A., 2006, "Robust and data-driven optimization: modern decision making under uncertainty", in *Tutorials in Operations Research, Models, Methods and Applications for Innovative Decision Making*, pp. 95-122, (INFORMS, Hanover, MD), 2006.
6. Brause, R., Langsdorf, T., and Hepp, M., 1999, "Neural data mining for credit card fraud detection", in *Tools with Artificial Intelligence*, and *Proc. of the 11th IEEE Int. Conf.*, 9-11 November 1999, pp. 103-106.
7. Brierley, P. and Batty B., 1999, "Data mining with neural networks—an applied example in understanding electricity consumption patterns", in *Knowledge Discovery and Data Mining,* edited by M.A. Bramer, pp. 182-188, (The Institute of Electrical Engineers, London), 1999.
8. Bunge, M., 1977, *Treatise on Basic Philosophy: Vol. 3: Ontology I: The Furniture of the World.* (Reidel, Boston, MA), 1977.
9. Bunge, M., 1979, *Treatise on Basic Philosophy: Vol. 4: Ontology II: A World of Systems.* (Reidel, Boston, MA), 1979.
10. Dhond, A., Gupta, A., and Vadhavkar, S., 2000, "Data mining techniques for optimizing inventories for electronic commerce", in *Proc. of the 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, August 2000.
11. Fu, M., 2002, "Optimization for Simulation: theory vs. practice", *INFORMS Journal on Computing*, 2002, **14**(3), pp.192-215.
12. Fu, M., Hill, S., 1997, "Optimization of discrete event systems via simultaneous perturbation stochastic approximation", *IIE Transactions*, 1997, **29**, pp. 233-243.
13. Garcia-Flores, R., Wang, X. Z., and Burgess, T. F., 2003, "Tuning inventory policy parameters in a small chemical company", *Journal of the Operations Research Society*, 2003, **54**, pp.350-361.
14. Ghiani, G., Laporte, G., and Musmanno, R., 2004, *Introduction to Logistics Systems Planning and Control*, (John Wiley & Sons Ltd, West Sussex, England), 2004.

15. Goldberg, D. E., 1989, *Genetic algorithms in search, optimization, and machine learning*, (Addison-Wesley, Reading), 1989.
16. Guyon, I. and Elisseeff, A., 2005, "An introduction to variable and feature selection". *Journal of Machine Learning Research*, 2005, **3**, pp. 1157-1182.
17. Hall, M.A., and Holmes, G., 2003, "Benchmarking attribute selection techniques for discrete class data mining". *IEEE Transactions on Knowledge and Data Engineering*, 2003, **15**(6), pp. 1437-1447.
18. Harris, F.W., 1913, "How many part to make at once", *Factory, The Magazine of Management*, 1913, **10**(2) (February 1913), 135-136, 152.
19. Howard, C.M. and Rayward-Smith, V.J., 1999, "Discovering knowledge from low-quality meteorological databases", in *Knowledge Discovery and Data Mining,* edited by M.A. Bramer, pp. 182-188, (The Institute of Electrical Engineers, London), 1999.
20. Kang, Y., and Gershwin, S. B., 2005, "Information inaccuracy in inventory systems: stock loss and stockout", *IIE Transactions*, 2005, **37**, pp. 843-859.
21. Kapuscinski, R. and Tayur, S., 1998, "A capacitated production-inventory model with periodic demand", *Operations Research*, 1998, **46**(6), 899-911.
22. Krajewski, L. J., Ritzman, L. P., and Malhotra, M. K., 2007, *Operations Management, Processes and Value Chains*, 8th ed., (Pearson Prentice Hall, New Jersey), 2007.
23. Lee, L. H., Teng, S., Chew, E. P., Lye, K. W., Lendermann, P., Karimi, I. A., Chen, Y., Koh, C. H., 2005, "Application of multi-objective simulation-optimization techniques to inventory management problems", *Proceedings of the 37th Winter Simulation Conference*, December 4-7, 2005, Orlando, FL.
24. Liu, H. and Setiono, R., 1996, "A probabilistic approach to feature selection: a filter solution", in *Proc. of the 13th Int'l Conference on Machine Learning*, pp. 319-327, 1996.
25. Liu, H. and Yu, L., 2005, "Toward integrating feature selection algorithms for classification and clustering". *IEEE Transactions on Knowledge and Data Engineering*, 2005, **17**(4), 491-502.
26. Muckstadt, J.A., 2005, *Analysis and Algorithms for Service Parts Supply Chains*, (Springer, New York), 2005.
27. Nahmias, S., 2005, *Production and Operations Analysis*, 5th ed., (McGraw-Hill, New York) 2005.
28. Palm, C., 1938, "Analysis of the Erlang traffic formulae for busy-signal arrangements", *Ericsson Technics*, 1938, 4, 39-58.
29. Papalexopoulos, A. D., Hao, S. Y. and Peng, T. M., 1994, An implementation of a neural network based load forecasting model for the EMS, *IEEE Transactions on Power Systems*, 1994, **9**(4), 1956-1962.
30. Paschalidis, I. C., Liu, Y., Cassandras, C. G., Panayiotou, C., 2004, "Inventory control for supply chains with service level constraints: a synergy between large deviations and perturbation analysis", *Annals of Operations Research*, 2004, 126, pp. 231-258.
31. Pfaff, B., 1999, "Count your parts, improving storeroom accuracy for maintenance customers", *IIE Solutions*, 1999, **31**(12), 29-31.

32. Piramuthu, S., 1999, "The Hausdorff distance measure for feature selection in learning applications", in *Proc. of the 32nd Annual Hawaii Int. Conf. on System Sciences* (*HICSS-32)*, 1999.
33. Pritsker, A. A. B., and O'Reilly, J. J., 1999, *Simulation with Visual SLAM and AweSim*, (John Wiley & Sons, New York), 1999.
34. Quinlan, R., 1993, *C4.5: Programs for Machine Learning*, (Morgan Kaufmann Publishers, San Mateo), 1993.
35. Sheffi, Y., 2005, *The Resilient Enterprise: Overcoming Vulnerability for Competitive Advantage*, (MIT Press, Cambridge, MA), 2005.
36. Simchi-Levi, D., Kaminsky, P., Simchi-Levi, E., 2004, *Managing the Supply Chain: The Definitive Guide for the Business Professional*, (McGraw-Hill, New York), 2004.
37. Sherbrooke, C. C., 2004, *Optimal Inventory Modeling of Systems Multi-Echelon Techniques, Second Edition*, (Kluwer, Norwell, MA), 2004.
38. Sherbrooke, C. C., 1984, *Estimation of the Variance-to-Mean Ratio for AFLC Recoverable Items*. Sherbrooke & Associates, Potomac, MD, 1984.
39. Schultz, K. L., Juran, D. C., Boudreau, J. W., McClain, J. O., and Thomas, L. J., 1998, "Modeling and worker motivation in JIT production systems", *Management Science*, 1998, **44**(12), Part 1 of 2, pp. 1595-1607.
40. Spall, J. C., 1992, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation", *IEEE Trans. On Automatic Control*, 1992, **37**(3), pp. 332-341.
41. Spall, J. C., 1998a, "Implementation of the simultaneous perturbation algorithm for stochastic optimization", *IEEE Trans. On Aerospace and Electronic Systems*, 1998, **34**(3), pp. 817-823.
42. Spall, J. C., 1998b, "An overview of the simultaneous perturbation method for efficient optimization", *John Hopkins APL Technical Digest*, 1998, **19**(4), pp. 482-492.
43. Spall, J. C., and Cristion, J. A., 1998, "Model-free control of nonlinear stochastic systems with discrete-time measurements", *IEEE Trans. On Automatic Control*, 1998, **43**(9), pp. 1198-1210.
44. Trafalis, T.B., Santosa, B. and Richman, M.B., 2005, "Learning networks for tornado forecasting: a Bayesian perspective", in *Data Mining VI: Data Mining, Text Mining and their Business Applications,* edited by A. Zanasi, C.A. Brebbia and N.F.F. Ebecken, pp. 5-14, 2005 (WITPress, Southhampton), 2005.
45. Vilalta, R. and Ma, S., 2002, "Predicting rare events in temporal domains", *Proc. of the 2002 IEEE Int. Conf. on Data Mining (ICDM 2002)*, 9-12 December 2002, pp.474 – 481.
46. Wai-Ho Au, C., Keith C.C., and Yao, X., 2003, "A novel evolutionary data mining algorithm with applications to churn prediction", *IEEE Transactions on Evolutionary Computation*, 2003, **7**(6), 532-545.
47. Wand, Y., Weber, R., 1990, "An ontological model of an information system", *IEEE Transactions on Software Engineering*, 1990, **16**(11), pp. 1282-1292.

48. Wang, Z., and Webb, G.I., 2002, "Comparison of lazy Bayesian rule, and tree-augmented Bayesian learning", *Proc. of the 2002 IEEE Int. Conf. on Data Mining (ICDM 2002)*, 9-12 December 2002, pp.490 – 497.
49. Webb, A., 1999, *Statistical Pattern Recognition*, (Arnold, London), 1999.
50. Wilson, A.J., Morgenstern, M.P., Pfahringer, B., and Leschi, C., 2004, "Data mining bread quality and process data in a plant bakery", in *Proc. of the 12th ICC Cereal and Bread Congress*, May 2004.
51. Witten, I.H. and Frank, E., 2005, *Data Mining: Practical Machine Learning Tools and Techniques -2nd ed.*, (Morgan Kaufmann, San Francisco), 2005.
52. Yohda, M.; Saito-Arita, M., Okada, A., Suzuki, R., and Kakemoto, Y., 2002, "Demand forecasting by the neural network with discrete fourier transform", in *Data Mining*, 2002, *Proc. of the ICDM* 2002 and 2002 *IEEE International Conference*, 9-12 December 2002, pp.779 – 782.

Appendix A

AWESIM simulation software code and diagrams for producing the four service part inventory demand archetypes

```
AweSim Input Translator, version 3.0
Copyright (C) 1999 Symix Systems, Inc.

Reading control MULTIMOD ...

   1 GEN,"MultiModTask","MULTI",1-FEB-2006,2000,YES,YES;
   2 LIMITS,50,50,50,50,50,50,300;
   3
EQUIVALENCE,{{AVGDMD,LL[5]},{STDVDMD,LL[6]},{DMDVARMFACTOR,LL[7]},{DMDVARSFACTOR,LL[8]
},{PARMRECS,LL[2]}};
   4
INTLC,{{SZ[1],"DDPeriodic.txt"},{SZ[2],"PERIODIC"},{LL[1],7},{XX[1],UNFRM(10,30,11)},{
LL[5],186},{LL[6],356},{LL[7],25},{LL[8],50},{LL[2],3}};
   5 NET;
   6 FIN;

MULTIMOD successfully read


Translated file PERIODIC successfully written
Reading network REGPRDNT - Pass 1...


REGPRDNT - Pass 1 successfully read

Reading network REGPRDNT - Pass 2...


REGPRDNT - Pass 2 successfully read

Reading network REGPRDNT - Pass 3...

   1 Start: CREATE,INF,0.0,,INF,1;
   2 ACTIVITY;
   3 ASSIGN,{{LTRIB[1],NINT(TRIAG(1,2,PARMRECS,1))}},1;
   4 ACTIVITY;
   5 READDMD: EVENT,1,1;
   6 ACTIVITY;
   7 SETLEVEL:
ASSIGN,{{XX[1],RLOGN(AVGDMD,STDVDMD,1)},{XX[2],RLOGN(XX[1]/DMDVARMFACTOR,XX[1]/DMDVARS
FACTOR,1)},{XX[3],0}},1;
   8 ACTIVITY;
   9 PERIODIC_NODE: CALLVSN,"REGDMD",,{XX[1],XX[2],XX[3]},1,"PERIODIC_DMD";
  10 ACTIVITY;
  11 TERMINATE,INF;
  12 OTPUT: ASSIGN,{{SZ[1],"DDPeriodic.txt"},{SZ[2],"PERIODIC"}},1;
  13 ACTIVITY,,,(XX[3]>=30);
  14 ACTIVITY,,,,"WRTCLASS";
  15 ASSIGN,{{SZ[2],"SPARSE"}},1;
  16 ACTIVITY;
  17 WRTCLASS: WRITE,SZ[1],NO,"%s\n",{SZ[2]},1;
  18 ACTIVITY;
  19 TERMINATE,INF;

REGPRDNT - Pass 3 successfully read

Reading subnetwork REGULAR ...

   1 VSN,REGDMD,{{PARTSNEEDED,DOUBLEVAL,Parts needed in a
repair},{VARIABILITY,DOUBLEVAL,How consistent is the demand},{ZERODMD,DOUBLEREF,The
number of zero demands}};
```

```
   2 LIMITSVSN,7,2,2,5,3,-1;
   3 ENTER2: ENTERVSN,PERIODIC_DMD,1;
   4 ACTIVITY;
   5 PERIODSN_GOON_1: GOON,2;
   6 ACTIVITY,,1,,"PERIODSN_GOON_1";
   7 ACTIVITY,,0;
   8 ASSIGN,{{SZINST[1],"DDPeriodic.txt"},{ANTRIB[1],NPSSN(1,1)}},1;
   9 ACTIVITY,,0,.25 <= ANTRIB[1];
  10 ACTIVITY,,,,"SSPSA1P_ASSIGN_1";
  11 ASSIGN,{{ANTRIB[2],NINT(RLOGN(PARTSNEEDED*ANTRIB[1],VARIABILITY,1))}},1;
  12 ACTIVITY,,,ANTRIB[2]!=0;
  13 ACTIVITY,,,ANTRIB[2]==0,"SUMZEROS";
  14 WRITEDMD: WRITE,SZINST[1],NO,"%5.0f,",{ANTRIB[2]},1;
  15 ACTIVITY;
  16 TERMINATE,36;
  17 SUMZEROS: ASSIGN,{{ZERODMD,ZERODMD+1}},1;
  18 ACTIVITY,,,,"WRITEDMD";
  19 SSPSA1P_ASSIGN_1: ASSIGN,{{ANTRIB[2],0}},1;
  20 ACTIVITY,,,,"SUMZEROS";

REGULAR successfully read

Warning:  no way to get to node in file REGPRDNT.net, line 12

Translated network file PERIODIC.TRN successfully written




AweSim Input Translator, version 3.0
Copyright (C) 1999 Symix Systems, Inc.

Reading control MULTIMOD ...

   1 GEN,"MultiModTask","MULTI",1-FEB-2006,2000,YES,YES;
   2 LIMITS,50,50,50,50,50,50,300;
   3
EQUIVALENCE,{{AVGDMD,LL[5]},{STDVDMD,LL[6]},{DMDVARMFACTOR,LL[7]},{DMDVARSFACTOR,LL[8]
},{PARMRECS,LL[2]}};
   4
INTLC,{{SZ[1],"DDPeriodic.txt"},{SZ[2],"PERIODIC"},{LL[1],7},{XX[1],UNFRM(10,30,11)},{
LL[5],186},{LL[6],356},{LL[7],25},{LL[8],50},{LL[2],3}};
   5 NET;
   6 FIN;

MULTIMOD successfully read


Translated file SEASON successfully written
Reading network SNSDMDNT - Pass 1...


SNSDMDNT - Pass 1 successfully read

Reading network SNSDMDNT - Pass 2...


SNSDMDNT - Pass 2 successfully read

Reading network SNSDMDNT - Pass 3...

   1 CREATE,INF,0.0,,INF,1;
   2 ACTIVITY;
   3 ASSIGN,{{LTRIB[1],NINT(TRIAG(1,2,PARMRECS,1))}},1;
   4 ACTIVITY;
   5 READDMD: EVENT,1,1;
   6 ACTIVITY;
   7 SETLEVEL2:
ASSIGN,{{XX[4],RLOGN(AVGDMD,STDVDMD,1)},{XX[5],RLOGN(XX[1]/DMDVARMFACTOR,XX[1]/DMDVARS
```

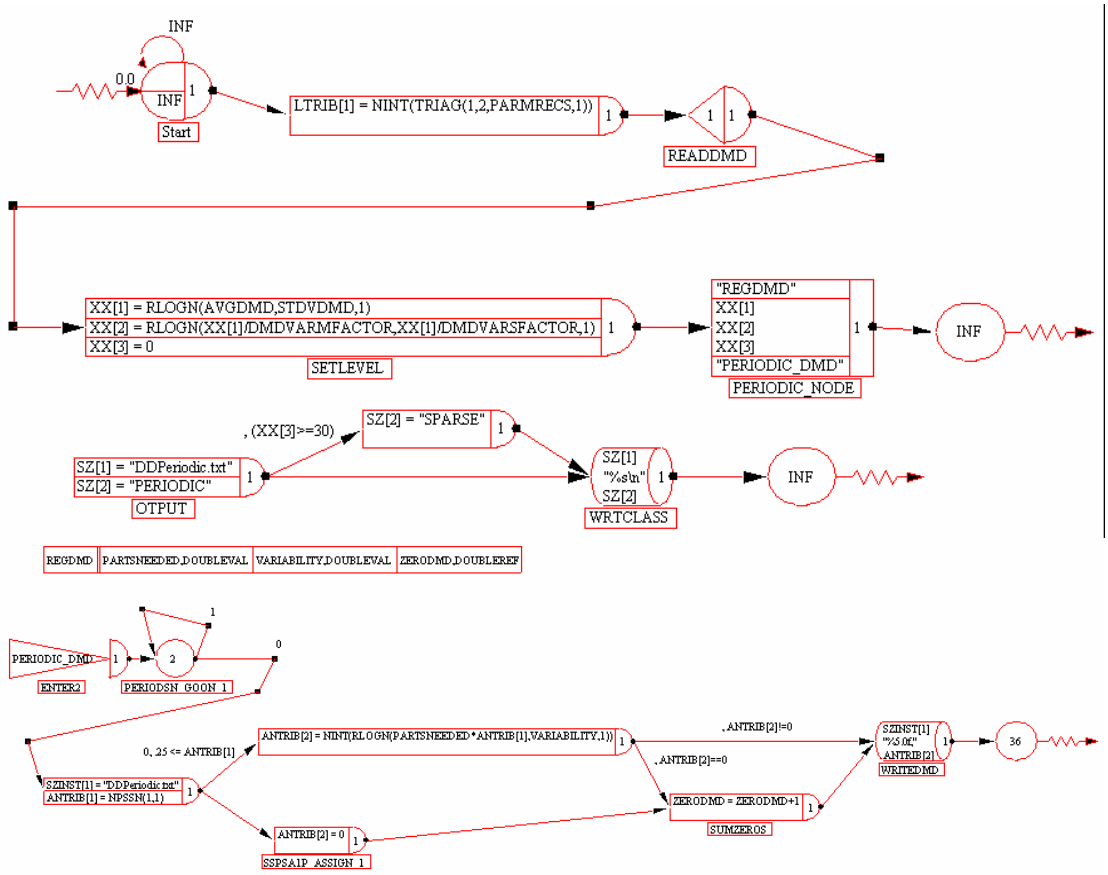
```
FACTOR,1)},{XX[6],0},{XX[7],NINT(UNFRM(3,12,1))},{XX[8],RLOGN(AVGDMD*UNFRM(.3,5.5,1),S
TDVDMD*UNFRM(.3,5.5,1),1)},{XX[9],0},{XX[1],DRAND(1)}},1;
   8 ACTIVITY,,,XX[1]<0.5;
   9 ACTIVITY,,,,"SEASONAL_NODE";
  10 ASSIGN,{{XX[8],XX[8]*-1.0}},1;
  11 ACTIVITY;
  12 SEASONAL_NODE:
CALLVSN,"SNSDMD",,{XX[4],XX[5],XX[6],XX[7],XX[8],XX[9]},1,"SEASONAL_DMD";
  13 ACTIVITY;
  14 TERMINATE,INF;
  15 OTPUT: ASSIGN,{{SZ[1],"DDSeasonal.txt"},{SZ[2],"SEASONAL"}},1;
  16 ACTIVITY,,,(XX[6]>=30);
  17 ACTIVITY,,,XX[9]<3 && XX[6]<30,"SNSDMDNT_ASSIGN_1";
  18 ACTIVITY,,,,"WRTCLASS";
  19 ASSIGN,{{SZ[2],"SPARSE"}},1;
  20 ACTIVITY;
  21 WRTCLASS: WRITE,SZ[1],NO,"%s\n",{SZ[2]},1;
  22 ACTIVITY;
  23 TERMINATE,INF;
  24 SNSDMDNT_ASSIGN_1: ASSIGN,{{SZ[2],"PERIODIC"}},1;
  25 ACTIVITY,,,,"WRTCLASS";

SNSDMDNT - Pass 3 successfully read

Reading subnetwork SEASONSN ...

   1 VSN,SNSDMD,{{PARTSNEEDED,DOUBLEVAL,Parts needed in a
repair},{VARIABILITY,DOUBLEVAL,How consistent is the demand},{ZERODMD,DOUBLEREF,The
number of zero demands},{SEASONALITY,DOUBLEVAL,The seasonality of the
demand},{DEMANDDELTA,DOUBLEVAL,The season induced demand
change},{SEASONS,DOUBLEREF,The number of season changes observed}};
   2 LIMITSVSN,7,2,2,5,3,-1;
   3 SEASON: ENTERVSN,SEASONAL_DMD,1;
   4 ACTIVITY;
   5 SNEX81A_GOON_1: GOON,2;
   6 ACTIVITY,,1,,"SNEX81A_GOON_1";
   7 ACTIVITY;
   8 ASSIGN,{{SZINST[1],"DDSeasonal.txt"},{ANTRIB[1],NPSSN(1,1)}},1;
   9 ACTIVITY,,,.75>ANTRIB[1],"SSPSA1P_ASSIGN_1";
  10 ACTIVITY,,0,.75 <= ANTRIB[1],"SEASONSN_ASSIGN_1";
  11 SSPSA1P_ASSIGN_1: ASSIGN,{{ANTRIB[2],0}},1;
  12 ACTIVITY,,,,"SUMZEROS";
  13 SUMZEROS: ASSIGN,{{ZERODMD,ZERODMD+1}},1;
  14 ACTIVITY,,,,"WRITEDMD";
  15 WRITEDMD: WRITE,SZINST[1],NO,"%5.0f,",{ANTRIB[2]},1;
  16 ACTIVITY;
  17 TERMINATE,36;
  18 SEASONSN_ASSIGN_1:
ASSIGN,{{ANTRIB[2],NINT(RLOGN(PARTSNEEDED*ANTRIB[1],VARIABILITY,1))}},1;
  19 ACTIVITY,,,ANTRIB[2]==0,"SUMZEROS";
  20
ACTIVITY,,0,(ANTRIB[2]!=0)&&(MOD(NINT(TNOW),NINT(SEASONALITY))==NINT(RNORM(0,0.1,1)));
  21 ACTIVITY,,,ANTRIB[2]!=0,"WRITEDMD";
  22
ASSIGN,{{ANTRIB[2],NINT(MAX(ANTRIB[2]+RNORM(DEMANDDELTA,ABS(DEMANDDELTA/25),1),0))},{S
EASONS,SEASONS+1}},1;
  23 ACTIVITY,,,ANTRIB[2]==0,"SUMZEROS";
  24 ACTIVITY,,,ANTRIB[2]!=0,"WRITEDMD";

SEASONSN successfully read

Warning:  no way to get to node in file SNSDMDNT.net, line 15

Translated network file SEASON.TRN successfully written



AweSim Input Translator, version 3.0
Copyright (C) 1999 Symix Systems, Inc.
```

```
Reading control MULTIMOD ...

   1 GEN,"MultiModTask","MULTI",1-FEB-2006,2000,YES,YES;
   2 LIMITS,50,50,50,50,50,50,300;
   3
EQUIVALENCE,{{AVGDMD,LL[5]},{STDVDMD,LL[6]},{DMDVARMFACTOR,LL[7]},{DMDVARSFACTOR,LL[8]
},{PARMRECS,LL[2]}};
   4
INTLC,{{SZ[1],"DDPeriodic.txt"},{SZ[2],"PERIODIC"},{LL[1],7},{XX[1],UNFRM(10,30,11)},{
LL[5],186},{LL[6],356},{LL[7],25},{LL[8],50},{LL[2],3}};
   5 NET;
   6 FIN;

MULTIMOD successfully read


Translated file LEVEL successfully written
Reading network LEVELNET - Pass 1...


LEVELNET - Pass 1 successfully read

Reading network LEVELNET - Pass 2...


LEVELNET - Pass 2 successfully read

Reading network LEVELNET - Pass 3...

   1 Start: CREATE,INF,0.0,,INF,2;
   2 ACTIVITY;
   3 ASSIGN,{{LTRIB[1],NINT(TRIAG(1,2,PARMRECS,1))}},1;
   4 ACTIVITY;
   5 READDMD: EVENT,1,1;
   6 ACTIVITY;
   7 SETLEVEL:
ASSIGN,{{XX[1],RLOGN(AVGDMD,STDVDMD,1)},{XX[2],RLOGN(XX[1]/DMDVARMFACTOR,XX[1]/DMDVARS
FACTOR,1)},{XX[3],0},{LL[1],NINT(UNFRM(4,30,1))},{XX[4],NINT(XX[1]*UNFRM(0.3,5.5,1))},
{XX[5],DRAND(1)},{LL[2],0}},1;
   8 ACTIVITY,,,,"PERIODIC_NODE";
   9 ACTIVITY,,,XX[5]<0.5,"LEVELNET_ASSIGN_2";
  10 PERIODIC_NODE:
CALLVSN,"LVLDMD",,{XX[1],XX[2],XX[3],LL[1],XX[4],LL[2]},1,"LEVELSHIFT_DMD";
  11 ACTIVITY;
  12 TERMINATE,INF;
  13 LEVELNET_ASSIGN_2: ASSIGN,{{XX[4],XX[4]*-1.0}},1;
  14 ACTIVITY,,,,"PERIODIC_NODE";
  15 OTPUT: ASSIGN,{{SZ[1],"DDLevel.txt"},{SZ[2],"LEVEL"}},1;
  16 ACTIVITY,,,(XX[3]>=30);
  17 ACTIVITY,,,XX[3]<30 && LL[2]<=2,"LEVELNET_ASSIGN_1";
  18 ACTIVITY,,,XX[3]<30 && LL[2]>2,"WRTCLASS";
  19 ASSIGN,{{SZ[2],"SPARSE"}},1;
  20 ACTIVITY;
  21 WRTCLASS: WRITE,SZ[1],NO,"%s\n",{SZ[2]},1;
  22 ACTIVITY;
  23 TERMINATE,INF;
  24 LEVELNET_ASSIGN_1: ASSIGN,{{SZ[2],"PERIODIC"}},1;
  25 ACTIVITY,,,,"WRTCLASS";

LEVELNET - Pass 3 successfully read

Reading subnetwork LEVELSN ...

   1 VSN,LVLDMD,{{PARTSNEEDED,DOUBLEVAL,Parts needed in a
repair},{VARIABILITY,DOUBLEVAL,How consistent is the demand},{ZERODMD,DOUBLEREF,The
number of zero demands},{SHIFTMONTH,LONGVAL,The month that the demand
changes},{DMDSHIFT,DOUBLEVAL,The demand change.},{SHIFTS,LONGREF,Number of level
shifts demands}};
   2 LIMITSVSN,7,2,2,5,3,-1;
   3 ENTER2: ENTERVSN,LEVELSHIFT_DMD,2;
   4 ACTIVITY,,,,"5";
```

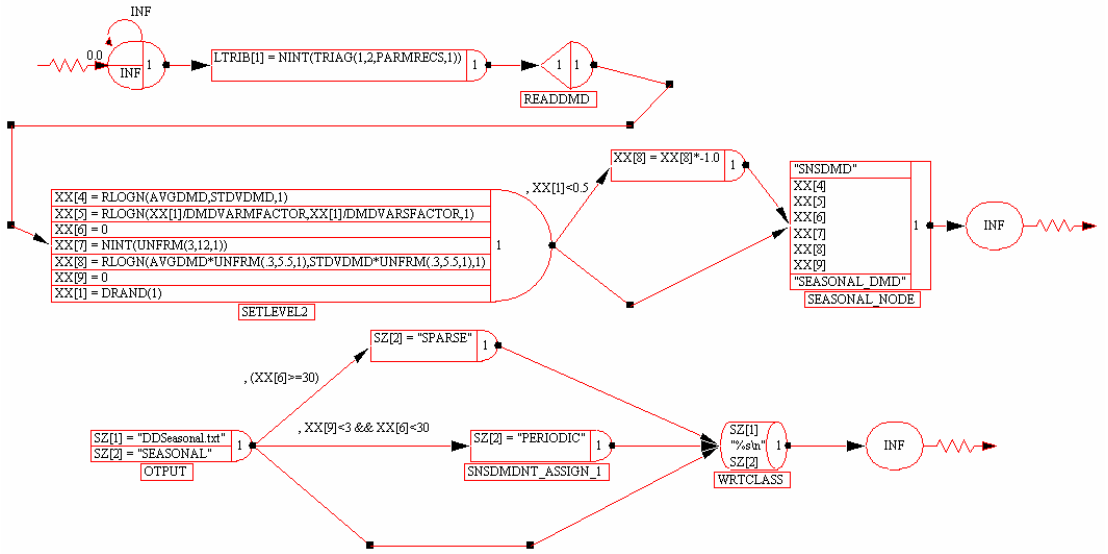
```
   5 ACTIVITY,,,,"PERIODSN_GOON_1";
   6 5: COLCT,1,SHIFTMONTH,"Level shift occurs",,,,1;
   7 ACTIVITY;
   8 6: COLCT,2,DMDSHIFT,"Demand Shift",,,,1;
   9 ACTIVITY;
  10 TERMINATE,INF;
  11 PERIODSN_GOON_1: GOON,2;
  12 ACTIVITY,,1,,"PERIODSN_GOON_1";
  13 ACTIVITY,,0;
  14 ASSIGN,{{SZINST[1],"DDLevel.txt"},{ANTRIB[1],NPSSN(1,1)}},1;
  15 ACTIVITY,,,1 > ANTRIB[1],"SSPSA1P_ASSIGN_1";
  16 ACTIVITY,,0,1 <= ANTRIB[1],"LEVELSN_ASSIGN_1";
  17 SSPSA1P_ASSIGN_1: ASSIGN,{{ANTRIB[2],0}},1;
  18 ACTIVITY,,,,"SUMZEROS";
  19 SUMZEROS: ASSIGN,{{ZERODMD,ZERODMD+1}},1;
  20 ACTIVITY,,,,"WRITEDMD";
  21 WRITEDMD: WRITE,SZINST[1],NO,"%5.0f,",{ANTRIB[2]},1;
  22 ACTIVITY;
  23 TERMINATE,36;
  24 LEVELSN_ASSIGN_1:
ASSIGN,{{ANTRIB[2],NINT(RLOGN(PARTSNEEDED*ANTRIB[1],VARIABILITY,1))}},1;
  25 ACTIVITY,,,ANTRIB[2]==0,"SUMZEROS";
  26 ACTIVITY,2,,TNOW>SHIFTMONTH;
  27 ACTIVITY,,,ANTRIB[2]!=0,"WRITEDMD";
  28 ASSIGN,{{ANTRIB[2],NINT(MAX(ANTRIB[2]+DMDSHIFT,0))},{SHIFTS,SHIFTS+1}},1;
  29 ACTIVITY,,,ANTRIB[2]==0,"SUMZEROS";
  30 ACTIVITY,1,,ANTRIB[2]>0,"WRITEDMD";

LEVELSN successfully read

Warning:  no way to get to node in file LEVELNET.net, line 15

Translated network file LEVEL.TRN successfully written




AweSim Input Translator, version 3.0
Copyright (C) 1999 Symix Systems, Inc.

Reading control MULTIMOD ...

   1 GEN,"MultiModTask","MULTI",1-FEB-2006,2000,YES,YES;
   2 LIMITS,50,50,50,50,50,50,300;
   3
EQUIVALENCE,{{AVGDMD,LL[5]},{STDVDMD,LL[6]},{DMDVARMFACTOR,LL[7]},{DMDVARSFACTOR,LL[8]
},{PARMRECS,LL[2]}};
   4
INTLC,{{SZ[1],"DDPeriodic.txt"},{SZ[2],"PERIODIC"},{LL[1],7},{XX[1],UNFRM(10,30,11)},{
LL[5],186},{LL[6],356},{LL[7],25},{LL[8],50},{LL[2],3}};
   5 NET;
   6 FIN;

MULTIMOD successfully read


Translated file SPARSE successfully written
Reading network SPRSNET - Pass 1...


SPRSNET - Pass 1 successfully read

Reading network SPRSNET - Pass 2...


SPRSNET - Pass 2 successfully read

Reading network SPRSNET - Pass 3...

   1 Start: CREATE,INF,0.0,,INF,1;
```

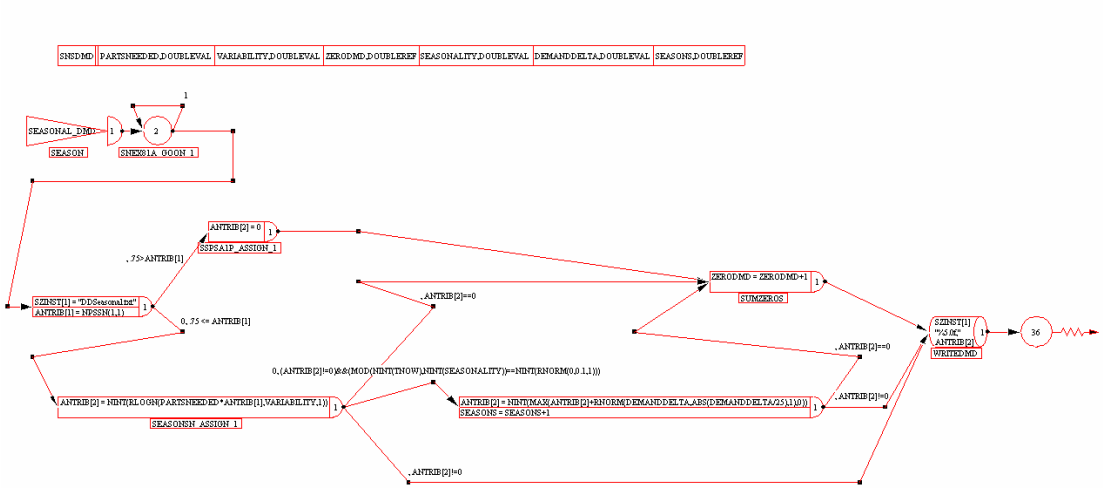
```
   2 ACTIVITY;
   3 ASSIGN,{{LTRIB[1],NINT(TRIAG(1,2,PARMRECS,1))}},1;
   4 ACTIVITY;
   5 READDMD: EVENT,1,1;
   6 ACTIVITY;
   7 SETLEVEL:
ASSIGN,{{XX[1],RLOGN(AVGDMD,STDVDMD,1)},{XX[2],RLOGN(XX[1]/DMDVARMFACTOR,XX[1]/DMDVARS
FACTOR,1)},{XX[3],0}},1;
   8 ACTIVITY;
   9 PERIODIC_NODE: CALLVSN,"SPRSDMD",,{XX[1],XX[2],XX[3]},1,"SPARSE_DMD";
  10 ACTIVITY;
  11 TERMINATE,INF;
  12 OTPUT: ASSIGN,{{SZ[1],"DDSparse.txt"},{SZ[2],"SPARSE"}},1;
  13 ACTIVITY,,,(XX[3]<30);
  14 ACTIVITY,,,,"WRTCLASS";
  15 ASSIGN,{{SZ[2],"PERIODIC"}},1;
  16 ACTIVITY;
  17 WRTCLASS: WRITE,SZ[1],NO,"%s\n",{SZ[2]},1;
  18 ACTIVITY;
  19 TERMINATE,INF;

SPRSNET - Pass 3 successfully read

Reading subnetwork SPARSESN ...

   1 VSN,SPRSDMD,{{PARTSNEEDED,DOUBLEVAL,Parts needed in a
repair},{VARIABILITY,DOUBLEVAL,How consistent is the demand},{ZERODMD,DOUBLEREF,The
number of zero demands}};
   2 LIMITSVSN,7,2,2,5,3,-1;
   3 ENTER2: ENTERVSN,SPARSE_DMD,1;
   4 ACTIVITY;
   5 PERIODSN_GOON_1: GOON,2;
   6 ACTIVITY,,1,,"PERIODSN_GOON_1";
   7 ACTIVITY,,0;
   8 ASSIGN,{{SZINST[1],"DDSparse.txt"},{ANTRIB[1],NINT(RNORM(0,8,1))}},1;
   9 ACTIVITY,,0,0 == ANTRIB[1];
  10 ACTIVITY,,,,"SSPSA1P_ASSIGN_1";
  11 ASSIGN,{{ANTRIB[2],NINT(RLOGN(PARTSNEEDED,VARIABILITY,1))}},1;
  12 ACTIVITY,,,ANTRIB[2]!=0;
  13 ACTIVITY,,,ANTRIB[2]==0,"SUMZEROS";
  14 WRITEDMD: WRITE,SZINST[1],NO,"%5.0f,",{ANTRIB[2]},1;
  15 ACTIVITY;
  16 TERMINATE,36;
  17 SUMZEROS: ASSIGN,{{ZERODMD,ZERODMD+1}},1;
  18 ACTIVITY,,,,"WRITEDMD";
  19 SSPSA1P_ASSIGN_1: ASSIGN,{{ANTRIB[2],0}},1;
  20 ACTIVITY,,,,"SUMZEROS";

SPARSESN successfully read

Warning:  no way to get to node in file SPRSNET.net, line 12

Translated network file SPARSE.TRN successfully written
```

Periodic demand network and subnetwork :

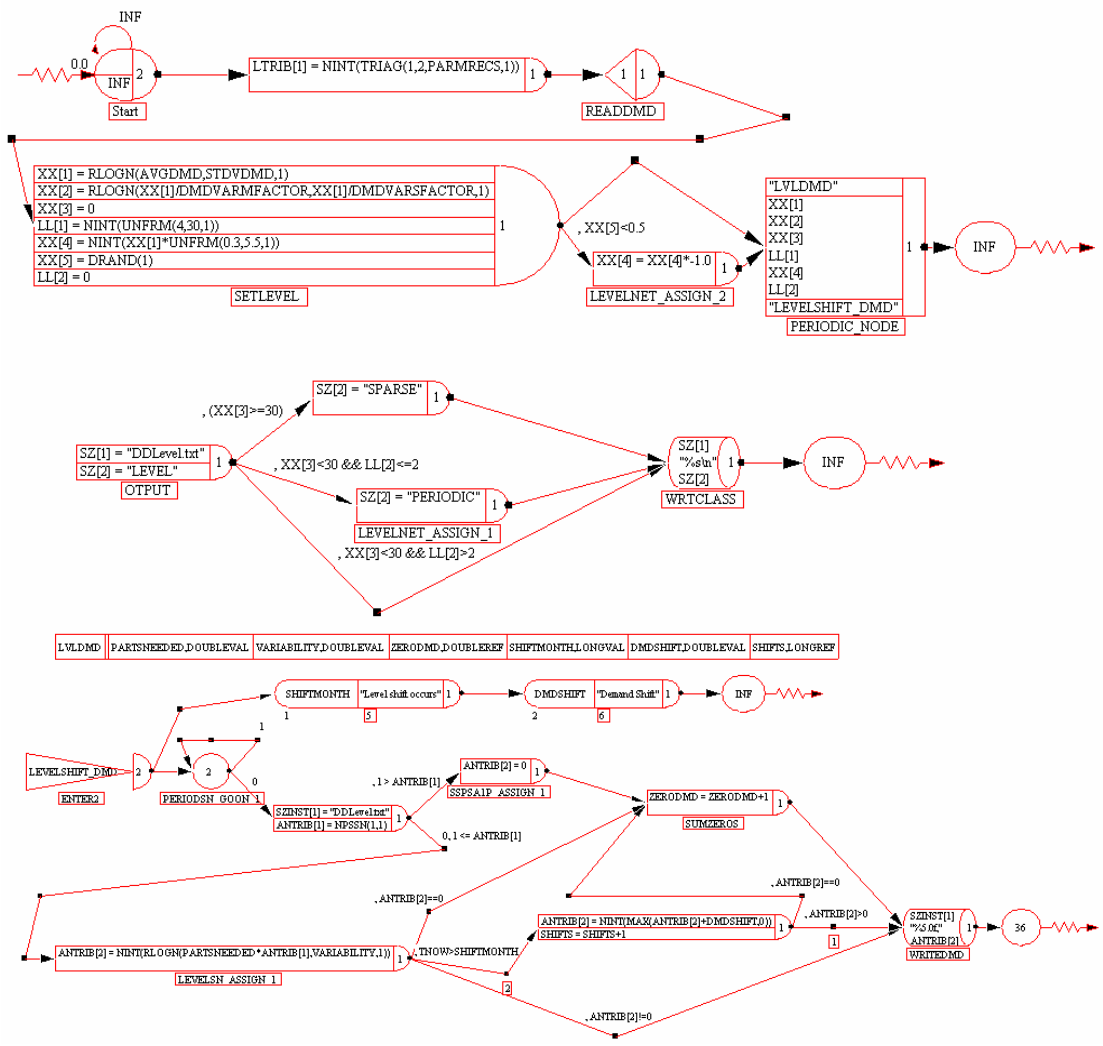


Seasonal demand network and subnetwork:

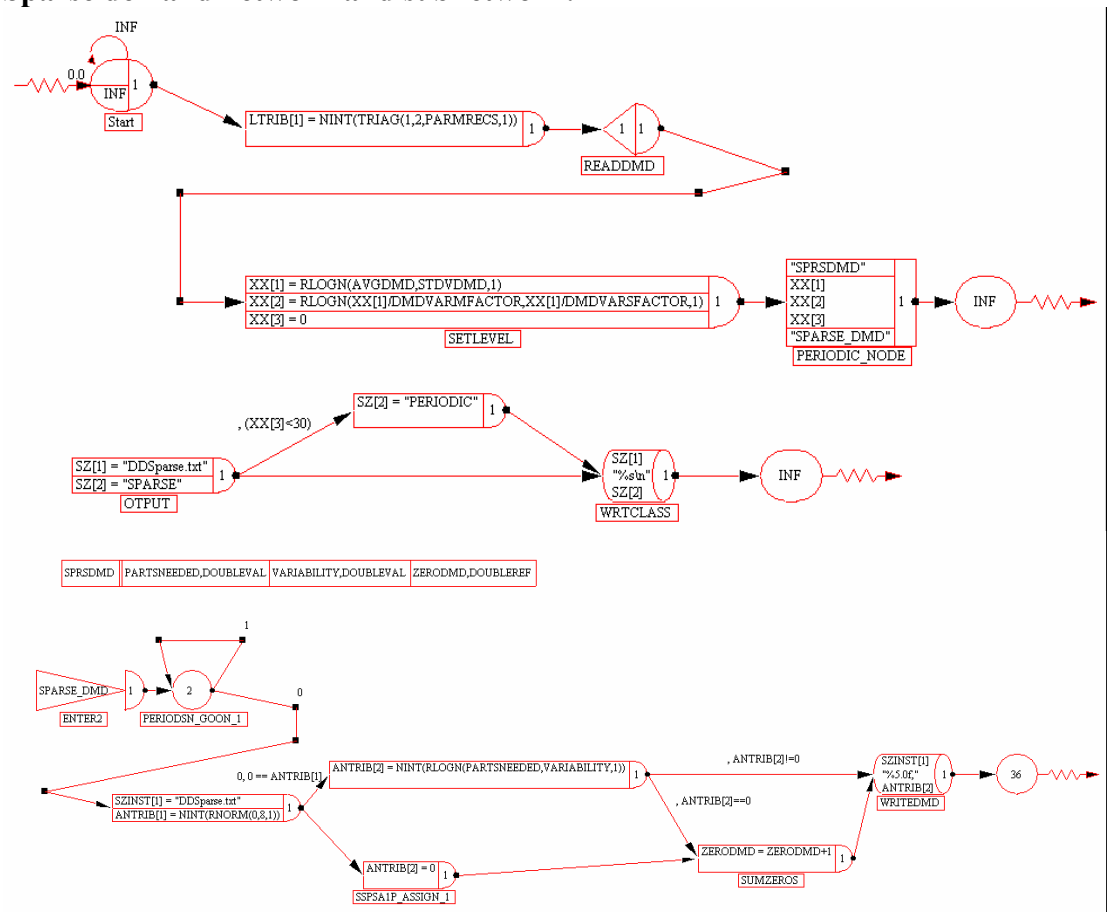




Level demand network and subnetwork:



Sparse demand network and subnetwork:



Appendix B

ANOVA Tables and diagnostics for the archetypal demand data mining experiments

Response: Success Rate
ANOVA for Selected Factorial Model
Analysis of variance table [Partial sum of squares]

Source	Sum of Squares	DF	Mean Square	F Value	Prob > F	
Model	33050.49	19	1739.50	35.86	< 0.0001	significant
A	*9786.61*	*3*	*3262.20*	*67.25*	*< 0.0001*	
B	*21871.64*	*4*	*5467.91*	*112.73*	*< 0.0001*	
AB	*1392.24*	*12*	*116.02*	*2.39*	*0.0077*	
Pure Error	6790.86	140	48.51			
Cor Total	39841.36	159				

The Model F-value of 35.86 implies the model is significant. There is only a 0.01% chance that a "Model F-Value" this large could occur due to noise.

Values of "Prob > F" less than 0.0500 indicate model terms are significant.
In this case A, B, AB are significant model terms.
Values greater than 0.1000 indicate the model terms are not significant.
If there are many insignificant model terms (not counting those required to support hierarchy),
model reduction may improve your model.

Std. Dev.	6.96	R-Squared	0.8296
Mean	53.65	Adj R-Squared	0.8064
C.V.	12.98	Pred R-Squared	0.7774
PRESS	8869.70	Adeq Precision	20.003

The "Pred R-Squared" of 0.7774 is in reasonable agreement with the "Adj R-Squared" of 0.8064.

"Adeq Precision" measures the signal to noise ratio. A ratio greater than 4 is desirable. Your
ratio of 20.003 indicates an adequate signal. This model can be used to navigate the design space.

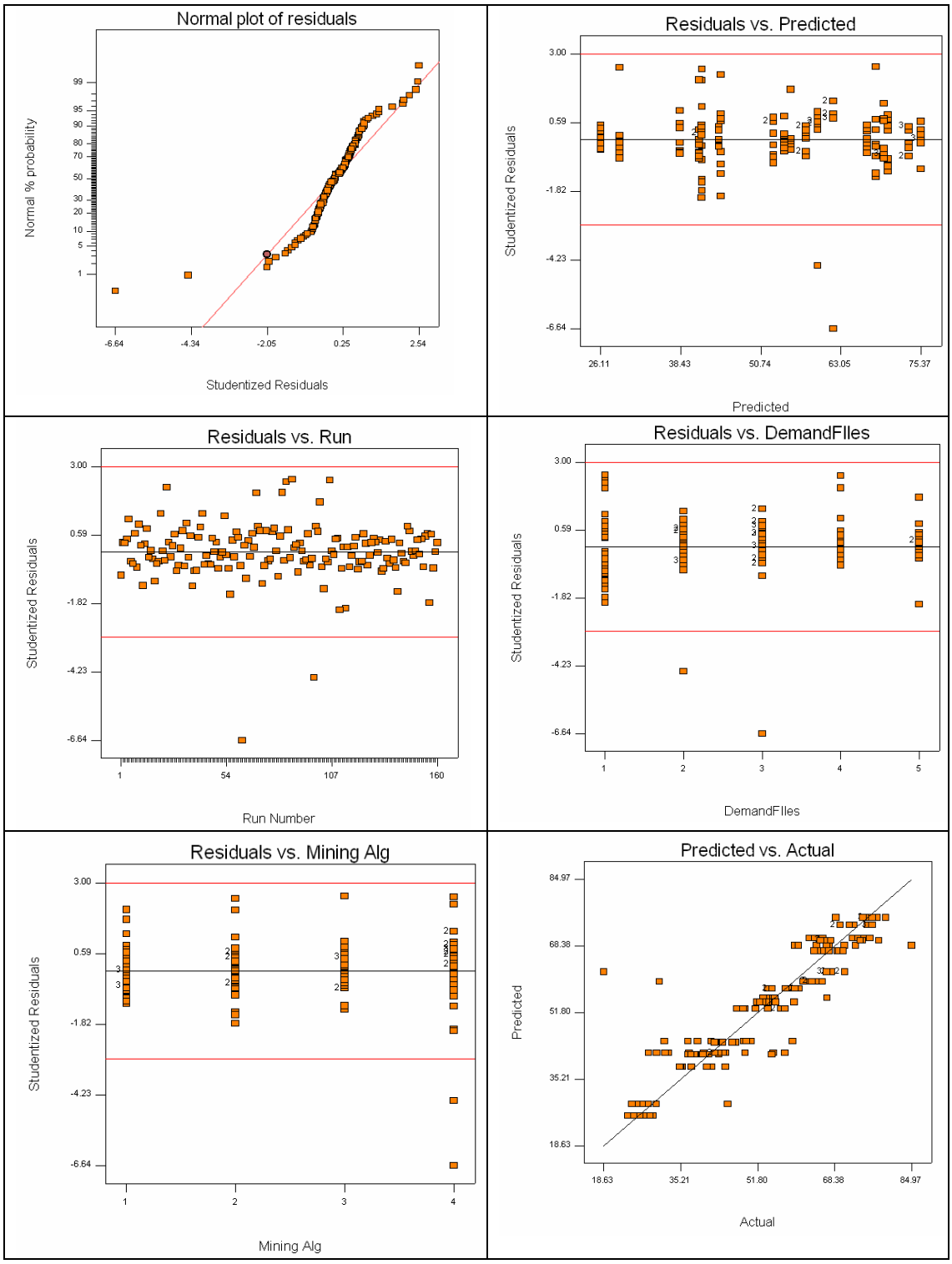


Figure 25: Success Rate ANOVA Diagnostics

Response: Kappa Statistic

ANOVA for Selected Factorial Model

Analysis of variance table [Partial sum of squares]

Source	Sum of Squares	DF	Mean Square	F Value	Prob > F	
Model	4.87	19	0.26	44.44	< 0.0001	significant
A	*3.12*	*3*	*1.04*	*180.34*	*< 0.0001*	
B	*1.39*	*4*	*0.35*	*60.11*	*< 0.0001*	
AB	*0.36*	*12*	*0.030*	*5.24*	*< 0.0001*	
Pure Error	0.81	140	5.764E-003			
Cor Total	5.67	159				

The Model F-value of 44.44 implies the model is significant. There is only a 0.01% chance that a "Model F-Value" this large could occur due to noise.

Values of "Prob > F" less than 0.0500 indicate model terms are significant.
In this case A, B, AB are significant model terms.
Values greater than 0.1000 indicate the model terms are not significant.
If there are many insignificant model terms (not counting those required to support hierarchy),
model reduction may improve your model.

Std. Dev.	0.076	R-Squared	0.8578
Mean	0.25	Adj R-Squared	0.8385
C.V.	30.68	Pred R-Squared	0.8142
PRESS	1.05	Adeq Precision	21.727

The "Pred R-Squared" of 0.8142 is in reasonable agreement with the "Adj R-Squared" of 0.8385.

"Adeq Precision" measures the signal to noise ratio. A ratio greater than 4 is desirable. Your
ratio of 21.727 indicates an adequate signal. This model can be used to navigate the design space.

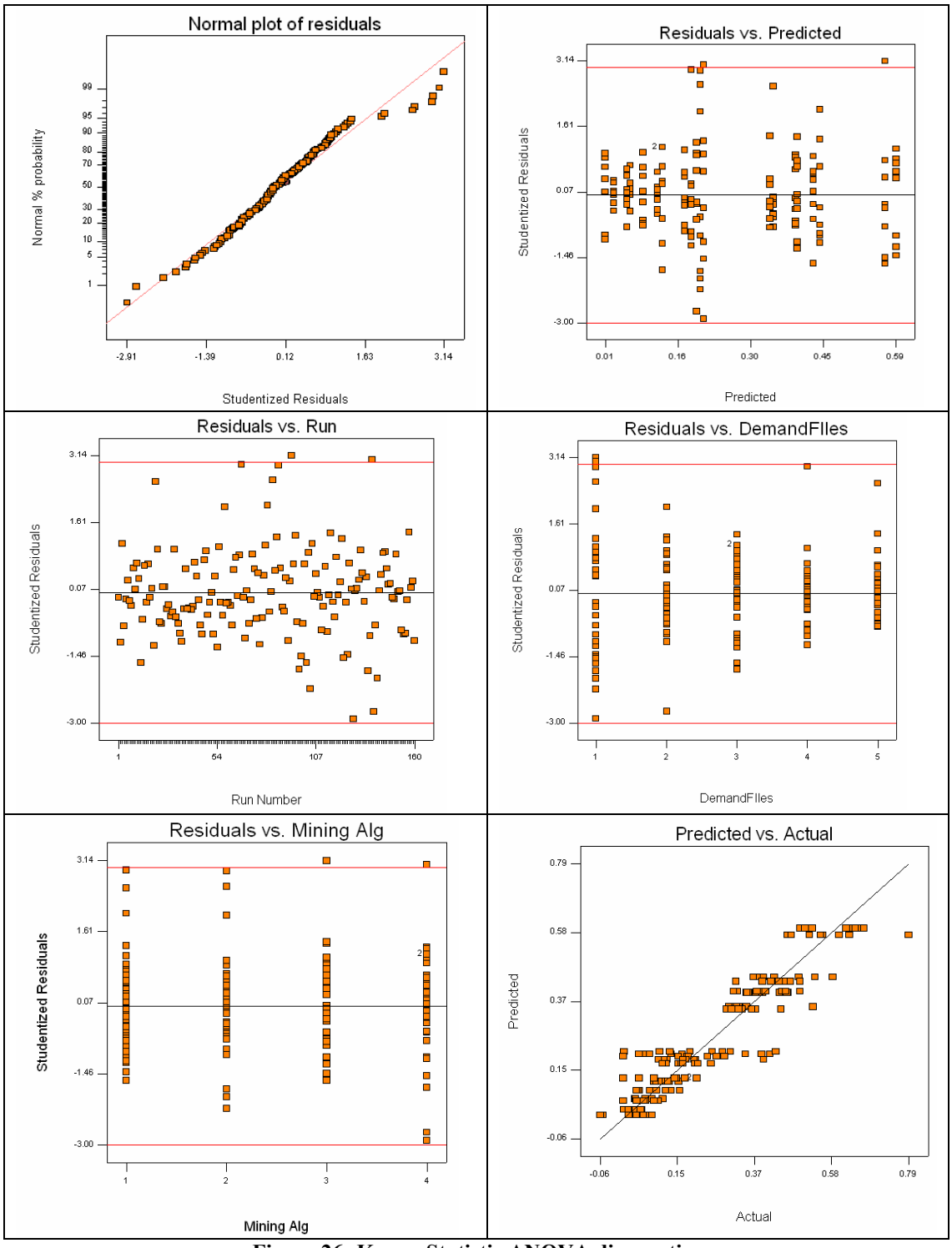


Figure 26: Kappa Statistic ANOVA diagnostics

Appendix C

AWESIM simulation software code and diagrams for modeling the multi-item service part inventory processes

```
AweSim Input Translator, version 3.0
Copyright (C) 1999 Symix Systems, Inc.

Reading control SPSAINIT ...

   1 ARRAY,1,3,{2,5,10};
   2 GEN,,,,10,YES,YES;
   3 LIMITS,38,37,5,16,11,1;
   4 PRIORITY,{{1,FIFO},{2,LVF(STRIB[1])}};
   5
EQUIVALENCE,{{NSN,STRIB[1]},{STOCK,ATRIB[1]},{Q,ATRIB[2]},{R,ATRIB[3]},{IDX,LL[4]},{UP
RICE,ATRIB[4]},{LEADTIME,ATRIB[5]},{LOC,LTRIB[1]},{CURDAY,XX[10]},{IDXREC,LL[6]},{INVP
OS,ATRIB[8]},{CD,LTRIB[2]},{DC,LTRIB[3]},{TIDC1,XX[17]},{TIDC2,XX[18]},{TIDC3,XX[19]},
{RECNUM,LTRIB[4]},{ORDERS,LTRIB[5]},{REVIEWS,LTRIB[6]},{BACKORDERS,ATRIB[14]},{BOSTART
,ATRIB[9]},{BODAYS,ATRIB[10]},{SPORDERS,LTRIB[7]},{REVIEWPERIOD,LTRIB[8]},{LASTORDERQ,
ATRIB[11]},{POLICY,LTRIB[10]},{STOCKLEVELSUM,ATRIB[12]},{ORDERQTYSUM,ATRIB[13]},{DDEST
,ATRIB[14]},{SDEST,ATRIB[15]},{DEMANDTYPE,LTRIB[11]},{ERRORRATE,ATRIB[16]}};
   6
EQUIVALENCE,{{QTY,ATRIB[1]},{DATE,ATRIB[2]},{SHIPTIME,ATRIB[3]},{DELTIME,ATRIB[4]},{IP
G,ATRIB[5]},{DUPRICE,ATRIB[6]},{STIMESAM,ATRIB[7]},{BOFLAG,LTRIB[1]},{TQ,ATRIB[8]},{DL
OC,LTRIB[2]},{DCD,LTRIB[3]},{DC1,LTRIB[4]},{F1,ATRIB[9]},{DC2,LTRIB[5]},{F2,ATRIB[10]}
,{DC3,LTRIB[6]},{F3,ATRIB[11]},{DCSEL,LTRIB[7]},{DCF,ATRIB[12]},{DLEADTIME,ATRIB[13]}}
;
   7 SEEDS,{{1783759,1,NO}};
   8
EQUIVALENCE,{{TRR,LL[1]},{TRS,LL[2]},{BO,LL[3]},{TIV,XX[5]},{BOPM,LL[5]},{POSOT,XX[8]}
,{COC,XX[9]},{TLTIME,XX[11]},{BONOSTK,LL[7]},{TEMPQ,XX[12]},{IDX2,LL[9]},{TEMPLOC,LL[1
0]},{VMITOT,LL[11]},{TRNPERMON,LL[12]},{TIVCD,XX[13]},{TRVCD,XX[14]},{TEMPCD,LL[13]},{
CDTRNPMON,LL[14]},{NUMDC,LL[15]},{MFCDC1,XX[20]},{MFCDC2,XX[21]},{MFCDC3,XX[22]},{TRSP
M,LL[16]},{STMON,LL[17]},{TRSHOLD,LL[18]},{IDX3,LL[19]},{TRR1,LL[20]},{TRR2,LL[21]},{T
RR3,LL[22]},{TRS1,LL[23]},{TRS2,LL[24]},{TRS3,LL[25]},{DLNI,LL[26]},{DLDT,LL[27]},{DLB
,LL[28]},{GBL,LL[29]},{NSEL1,LL[30]},{NSEL2,LL[31]},{NSEL3,LL[32]},{CASER,SZ[2]},{CASE
INV,SZ[3]},{CASEOUT,SZ[4]},{TSTOCK,XX[23]},{TINVPOS,XX[24]},{TQTY,XX[25]},{DIRECTION,L
L[35]},{COSTOUT,SZ[5]},{NREVIEWS,XX[30]},{NORDERS,XX[29]},{NBACKORDERS,XX[37]},{NBODAY
S,XX[31]},{NSPORDERS,LL[37]},{DELTA1,XX[32]},{DELTA2,XX[33]},{DELTA3,XX[34]},{MCOST,XX
[35]},{TAVGSTOCKLVL,LL[34]},{TAVGINVVALUE,XX[28]},{TORDERQTYVALUE,XX[36]},{TREVIEWERRO
R,XX[38]},{TREVIEWLEVEL,LL[36]}};
   9 INTLC,{{COC,0.12},{NUMDC,3}};
  10
INTLC,{{CASER,"DEMAND.TXT"},{CASEINV,"INVENTORY.TXT"},{CASEOUT,"COSTOUT_INIT.TXT"},{NU
MDC,1}};
  11 INITIALIZE,0.0,1096,YES,,NO;
  12 TIMST,1,TIV,"Total Inventory per Month",0,0.0,1.0;
  13 TIMST,4,TIVCD,"CostDriver Inventory per Month",0,0.0,1.0;
  14 TIMST,3,BO,"BO per month",0,0.0,1.0;
  15 TIMST,2,COC*TIV,"Cost of Cap",0,0.0,1.0;
  16 TIMST,5,TIDC1,"TotInvDC1",0,0.0,1.0;
  17 TIMST,6,TIDC2,"TotInvDC2",0,0.0,1.0;
  18 TIMST,7,TIDC3,"TotInvDC3",0,0.0,1.0;
  19 MONTR,SUMMARY,30.005,30;
  20 MONTR,CLEAR,30.01,30;
  21 NET;
  22 FIN;

SPSAINIT successfully read


Translated file INIT successfully written
Reading network DEMAND5 - Pass 1...
```

```
DEMAND5 - Pass 1 successfully read

Reading network LDINV1ST - Pass 1...

LDINV1ST - Pass 1 successfully read

Reading network RECS1ST - Pass 1...

RECS1ST - Pass 1 successfully read

Reading network REVIEWRQ - Pass 1...

REVIEWRQ - Pass 1 successfully read

Reading network REVIEWSS - Pass 1...

REVIEWSS - Pass 1 successfully read

Reading network STATS1ST - Pass 1...

STATS1ST - Pass 1 successfully read

Reading network DEMAND5 - Pass 2...

DEMAND5 - Pass 2 successfully read

Reading network LDINV1ST - Pass 2...

LDINV1ST - Pass 2 successfully read

Reading network RECS1ST - Pass 2...

RECS1ST - Pass 2 successfully read

Reading network REVIEWRQ - Pass 2...

REVIEWRQ - Pass 2 successfully read

Reading network REVIEWSS - Pass 2...

REVIEWSS - Pass 2 successfully read

Reading network STATS1ST - Pass 2...

STATS1ST - Pass 2 successfully read

Reading network DEMAND5 - Pass 3...

   1 ;DEMAND FOR INVENTORY
   2 DEMAND: GOON,1;
   3 ACTIVITY;
   4 Samp_Shiptime: ASSIGN,{{DELTIME,DELTIME*0.9},{STIMESAM,SHIPTIME*0.9}},1;
   5 ACTIVITY,,,,"SELDC";
   6 DEMAND2:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL},{XX[16],DCF}},1;
   7 ACTIVITY;
   8 FINDAR,1,NSN == SZ[1] && STOCK >= XX[1] && DC ==
XX[15],1,FORWARD,1,,{{STOCK,STOCK - XX[1]},{INVPOS,INVPOS - XX[1]},{TEMPCD,CD}},IDX,1;
   9 ACTIVITY,,,IDX == 0,"BKORD";
```

```
  10 ACTIVITY,,,IDX > 0,"DEMAND5_ASSIGN_2";
  11 BKORD:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL},{XX[16],DCF}},1;
  12 ACTIVITY;
  13 CHECK: FINDAR,1,NSN == SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{TEMPLOC,LOC},{TLTIME,LEADTIME},{INVPOS,INVPOS -
XX[1]},{TEMPCD,CD},{BACKORDERS,BACKORDERS+XX[1]},{BOSTART,TNOW}},IDX,1;
  14 ACTIVITY;
  15 DEMAND_ASSIGN_3:
ASSIGN,{{DLOC,TEMPLOC},{DLEADTIME,TLTIME},{DCD,TEMPCD},{BOFLAG,1},{BONOSTK,BONOSTK +
1}},1;
  16 ACTIVITY,,,,"DEMAND5_ASSIGN_1";
  17 DEMAND5_ASSIGN_1:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL}},1;
  18 ACTIVITY;
  19 DEMAND5_FINDAR_1: FINDAR,1,NSN == SZ[1] && DC == XX[15] && Q ==
0,1,FORWARD,1,,{{SPORDERS,SPORDERS+1}},IDX,1;
  20 ACTIVITY,,,IDX > 0,"SPORD";
  21 ACTIVITY,,,IDX == 0;
  22 BORDER: EVENT,2,1;
  23 ACTIVITY;
  24 TERMINATE,INF;
  25 DEMAND5_ASSIGN_2: ASSIGN,{{DCD,TEMPCD}},2;
  26 ACTIVITY;
  27 HAVESTOCK:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL},{XX[16],DCF}},1;
  28 ACTIVITY;
  29 DEMAND4_FINDAR_1: FINDAR,1,NSN == SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{TEMPLOC,LOC},{TLTIME,LEADTIME}},IDX,2;
  30 ACTIVITY;
  31 DEMAND4_ASSIGN_5: ASSIGN,{{DLOC,TEMPLOC},{DLEADTIME,TLTIME}},1;
  32 ACTIVITY,,STIMESAM,,"DEMAND_ASSIGN_1";
  33 DEMAND_ASSIGN_1: ASSIGN,{{TIV,TIV - (QTY*DUPRICE)},{TRVCD,TRVCD +
QTY*DUPRICE*DCD},{XX[6],XX[6] + QTY*DUPRICE},{TIVCD,TIVCD - QTY*DUPRICE*DCD}},1;
  34 ACTIVITY,,,DCSEL== 1;
  35 ACTIVITY,,,DCSEL == 2,"LDINV2_ASSIGN_1";
  36 ACTIVITY,,,DCSEL == 3,"LDINV2_ASSIGN_2";
  37 ASSIGN,{{TIDC1,TIDC1 - (QTY*DUPRICE)},{MFCDC1,MFCDC1 + DCF}},1;
  38 ACTIVITY;
  39 DEMAND_COLCT_1: COLCT,7,TNOW-DATE,"Time To Ship",5,1,1,1;
  40 ACTIVITY,,,IPG==1;
  41 ACTIVITY,,,IPG==2,"DEMAND4_COLCT_5";
  42 ACTIVITY,,,IPG==3,"DEMAND4_COLCT_6";
  43 COLCT,20,TNOW - DATE,"TTS_IPG1",,,,1;
  44 ACTIVITY,,DELTIME;
  45 DEMAND4_COLCT_4: COLCT,18,TNOW - DATE,"Time To Deliver",5,1,1,1;
  46 ACTIVITY,,,IPG==1;
  47 ACTIVITY,,,IPG == 2,"DEMAND4_COLCT_2";
  48 ACTIVITY,,,IPG == 3,"DEMAND4_COLCT_3";
  49 COLCT,23,TNOW - DATE,"TTD-IPG1",,,,1;
  50 ACTIVITY;
  51 DEMAND4_GOON_1: GOON,1;
  52 ACTIVITY,,,TNOW - DATE <= ARRAY[1,IPG];
  53 ACTIVITY,,,TNOW - DATE > ARRAY[1,IPG],"DEMAND_ASSIGN_2";
  54 ACTIVITY,,,,"DEMAND4_EVENT_1";
  55 SHIPPED: ASSIGN,{{TRS,TRS + 1}},1;
  56 ACTIVITY,,,IPG == 1;
  57 ACTIVITY,,,IPG == 2,"DEMAND4_ASSIGN_1";
  58 ACTIVITY,,,IPG == 3,"DEMAND4_ASSIGN_2";
  59 ASSIGN,{{TRS1,TRS1 + 1}},1;
  60 ACTIVITY;
  61 DEMAND4_TERMINATE_1: TERMINATE,INF;
  62 DEMAND4_ASSIGN_1: ASSIGN,{{TRS2,TRS2 + 1}},1;
  63 ACTIVITY,,,,"DEMAND4_TERMINATE_1";
  64 DEMAND4_ASSIGN_2: ASSIGN,{{TRS3,TRS3 + 1}},1;
  65 ACTIVITY,,,,"DEMAND4_TERMINATE_1";
  66 DEMAND_ASSIGN_2: ASSIGN,{{BO,BO + 1}},1;
  67 ACTIVITY;
```

```
  68 GOON,1;
  69 ACTIVITY,,,BOFLAG == 1 && ((DELTIME + STIMESAM) > ARRAY[1,IPG]);
  70 ACTIVITY,,,BOFLAG == 1,"DEMAND4_ASSIGN_3";
  71 ACTIVITY,,,((DELTIME + STIMESAM) > ARRAY[1,IPG]),"DEMAND4_ASSIGN_4";
  72 ACTIVITY,,,,"DEMAND4_EVENT_2";
  73 ASSIGN,{{DLB,DLB + 1}},1;
  74 ACTIVITY,,,,"DEMAND4_TERMINATE_1";
  75 DEMAND4_ASSIGN_3: ASSIGN,{{DLNI,DLNI + 1}},1;
  76 ACTIVITY,,,,"DEMAND4_TERMINATE_1";
  77 DEMAND4_ASSIGN_4: ASSIGN,{{DLDT,DLDT + 1}},1;
  78 ACTIVITY,,,,"DEMAND4_TERMINATE_1";
  79 DEMAND4_EVENT_2: EVENT,7,1;
  80 ACTIVITY;
  81 TERMINATE,1;
  82 DEMAND4_EVENT_1: EVENT,6,1;
  83 ACTIVITY;
  84 TERMINATE,1;
  85 DEMAND4_COLCT_2: COLCT,24,TNOW - DATE,"TTD-IPG2",,,,1;
  86 ACTIVITY,,,,"DEMAND4_GOON_1";
  87 DEMAND4_COLCT_3: COLCT,25,TNOW - DATE,"TTD-IPG3",,,,1;
  88 ACTIVITY,,,,"DEMAND4_GOON_1";
  89 DEMAND4_COLCT_5: COLCT,21,TNOW - DATE,"TTS_IPG2",,,,1;
  90 ACTIVITY,,DELTIME,,"DEMAND4_COLCT_4";
  91 DEMAND4_COLCT_6: COLCT,22,TNOW - DATE,"TTS_IPG3",,,,1;
  92 ACTIVITY,,DELTIME,,"DEMAND4_COLCT_4";
  93 LDINV2_ASSIGN_1: ASSIGN,{{TIDC2,TIDC2 - (QTY*DUPRICE)},{MFCDC2,MFCDC2 + DCF}},1;
  94 ACTIVITY,,,,"DEMAND_COLCT_1";
  95 LDINV2_ASSIGN_2: ASSIGN,{{TIDC3,TIDC3 - (QTY*DUPRICE)},{MFCDC3,MFCDC3 + DCF}},1;
  96 ACTIVITY,,,,"DEMAND_COLCT_1";
  97 ;CHECK FOR STOCK ETC UPDATE
  98 ;BO Order-Demand entity only
  99 SPORD: GOON,1;
 100 ACTIVITY;
 101
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL},{XX[16],DCF}},1;
 102 ACTIVITY;
 103 FINDAR,1,STRIB[1] == SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{TLTIME,LEADTIME},{ORDERQTYSUM,ORDERQTYSUM+XX[1]}},IDX,2;
 104 ACTIVITY,,RLOGN(TLTIME,0.1,1),IDX > 0;
 105 ACTIVITY,,,IDX > 0,"DEMAND4_COLCT_1";
 106
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL},{XX[16],DCF}},1;
 107 ACTIVITY;
 108 FINDAR,1,NSN == SZ[1] && DC == XX[15],1,FORWARD,1,,{{BODAYS,BODAYS+(TNOW-
BOSTART)},{INVPOS,INVPOS+XX[1]}},IDX,1;
 109 ACTIVITY;
 110 ASSIGN,{{BONOSTK,BONOSTK - 1},{TRVCD,TRVCD + QTY*DUPRICE*DCD},{XX[6],XX[6] +
QTY*DUPRICE}},1;
 111 ACTIVITY,,,DCSEL== 1;
 112 ACTIVITY,,,DCSEL == 2,"DEMAND_ASSIGN_10";
 113 ACTIVITY,,,DCSEL == 3,"DEMAND_ASSIGN_9";
 114 ASSIGN,{{MFCDC1,MFCDC1 + DCF}},1;
 115 ACTIVITY;
 116 DEMAND_COLCT_2: GOON,1;
 117 ACTIVITY,,STIMESAM,,"DEMAND_COLCT_1";
 118 DEMAND_ASSIGN_10: ASSIGN,{{MFCDC2,MFCDC2 + DCF}},1;
 119 ACTIVITY,,,,"DEMAND_COLCT_2";
 120 DEMAND_ASSIGN_9: ASSIGN,{{MFCDC3,MFCDC3 + DCF}},1;
 121 ACTIVITY,,,,"DEMAND_COLCT_2";
 122 DEMAND4_COLCT_1: COLCT,19,TLTIME,"Lead Time",,,,1;
 123 ACTIVITY;
 124 TERMINATE,INF;
 125 ;Select DC to use
 126 SELDC:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
1},{XX[16],F1}},1;
 127 ACTIVITY;
 128 FINDAR,1,NSN == SZ[1] && STOCK >= XX[1] && DC == XX[15],1,FORWARD,1,,,IDX,1;
 129 ACTIVITY,,,IDX > 0;
```

```
 130 ACTIVITY,,,IDX == 0 && NUMDC > 1,"DEMAND_ASSIGN_7";
 131 ACTIVITY,,,IDX == 0 && NUMDC == 1,"DEMAND_ASSIGN_8";
 132 ASSIGN,{{DCSEL,DC1},{DCF,F1},{NSEL1,NSEL1 + 1}},1;
 133 ACTIVITY,,,,"DEMAND2";
 134 DEMAND_ASSIGN_7:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
2},{XX[16],F2}},1;
 135 ACTIVITY;
 136 FINDAR,1,NSN == SZ[1] && STOCK >= XX[1] && DC == XX[15],1,FORWARD,1,,,IDX,1;
 137 ACTIVITY,,,IDX > 0;
 138 ACTIVITY,,,IDX == 0 && NUMDC > 2,"DEMAND_ASSIGN_5";
 139 ACTIVITY,,,IDX == 0 && NUMDC == 2,"DEMAND_ASSIGN_6";
 140 ASSIGN,{{DCSEL,DC2},{DCF,F2},{NSEL2,NSEL2 + 1}},1;
 141 ACTIVITY,,,,"DEMAND2";
 142 DEMAND_ASSIGN_5:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
3},{XX[16],F3}},1;
 143 ACTIVITY;
 144 FINDAR,1,NSN == SZ[1] && STOCK >= XX[1] && DC == XX[15],1,FORWARD,1,,,IDX,1;
 145 ACTIVITY,,,IDX > 0;
 146 ACTIVITY,,,IDX == 0,"DEMAND_ASSIGN_4";
 147 ASSIGN,{{DCSEL,DC3},{DCF,F3},{NSEL3,NSEL3 + 1}},1;
 148 ACTIVITY,,,,"DEMAND2";
 149 DEMAND_ASSIGN_4: ASSIGN,{{DCSEL,DC1},{DCF,F1},{NSEL1,NSEL1 + 1}},1;
 150 ACTIVITY,,,,"DEMAND2";
 151 DEMAND_ASSIGN_6: ASSIGN,{{DCSEL,DC1},{DCF,F1},{NSEL1,NSEL1 + 1}},1;
 152 ACTIVITY,,,,"DEMAND2";
 153 DEMAND_ASSIGN_8: ASSIGN,{{DCSEL,DC1},{DCF,F1},{NSEL1,NSEL1+1}},1;
 154 ACTIVITY,,,,"DEMAND2";

DEMAND5 - Pass 3 successfully read

Reading network LDINV1ST - Pass 3...

   1 ;Load Inventory Files
   2 CREATE,INF,0.0,,1,1;
   3 ACTIVITY;
   4 NET1_READ_1:
READ,CASEINV,YES,IDX,,{RECNUM,NSN,STOCK,Q,R,REVIEWPERIOD,POLICY,UPRICE,LEADTIME,LOC,CD
,DC,DDEST,SDEST,DEMANDTYPE,ERRORRATE},1;
   5 ACTIVITY,,,IDX > 0;
   6 ACTIVITY,,,IDX == 0,"NET1_TERMINATE_1";
   7
ASSIGN,{{STOCK,MAX(STOCK,0)},{INVPOS,STOCK},{ORDERS,0},{REVIEWS,0},{BACKORDERS,0},{STO
CKLEVELSUM,0},{SPORDERS,0},{BODAYS,0},{ORDERQTYSUM,0}},1;
   8 ACTIVITY;
   9 LDINVSP_EVENT_1: EVENT,1,1;
  10 ACTIVITY,,,LOC > 1;
  11 ACTIVITY,,,LOC == 1,"LDINV_ASSIGN_6";
  12 LDINV_ASSIGN_1: ASSIGN,{{TIV,TIV + MAX(UPRICE*STOCK,0)}},1;
  13 ACTIVITY,,,DC == 1;
  14 ACTIVITY,,,DC == 2,"LDINV2_ASSIGN_9";
  15 ACTIVITY,,,DC == 3,"LDINV2_ASSIGN_7";
  16 ACTIVITY,,,DC < 1 || DC > 3,"LDINV2_EVENT_1";
  17 ASSIGN,{{TIDC1,TIDC1 + MAX(UPRICE*STOCK,0)}},1;
  18 ACTIVITY;
  19 LDINV2_GOON_1: GOON,2;
  20 ACTIVITY;
  21 ACTIVITY,,,,"LDINVSP_GOON_1";
  22 GOON,1;
  23 ACTIVITY,,,CD == 0,"NET1_READ_1";
  24 ACTIVITY,,,CD == 1;
  25 LDINV_ASSIGN_3: ASSIGN,{{TIVCD,TIVCD + MAX(UPRICE*STOCK,0)}},1;
  26 ACTIVITY,,,,"NET1_READ_1";
  27 LDINVSP_GOON_1: GOON,1;
  28 ACTIVITY,,UNFRM(0,(REVIEWPERIOD-1)),POLICY <= 4,"REVIEWRQ";
  29 ACTIVITY,,UNFRM(0,(REVIEWPERIOD-1)),POLICY >= 5,"REVIEWSS";
  30 ACTIVITY;
  31 PolicyError: EVENT,9,1;
  32 ACTIVITY;
  33 TERMINATE,INF;
```

```
  34 LDINV2_ASSIGN_9: ASSIGN,{{TIDC2,TIDC2 + MAX(UPRICE*STOCK,0)}},1;
  35 ACTIVITY,,,,"LDINV2_GOON_1";
  36 LDINV2_ASSIGN_7: ASSIGN,{{TIDC3,TIDC3 + MAX(UPRICE*STOCK,0)}},1;
  37 ACTIVITY,,,,"LDINV2_GOON_1";
  38 LDINV2_EVENT_1: EVENT,4,1;
  39 ACTIVITY;
  40 ERROR_wrong_DC: TERMINATE,1;
  41 LDINV_ASSIGN_6: ASSIGN,{{VMITOT,VMITOT + 1}},1;
  42 ACTIVITY,,,,"LDINV_ASSIGN_1";
  43 NET1_TERMINATE_1: TERMINATE,INF;

LDINV1ST - Pass 3 successfully read

Reading network RECS1ST - Pass 3...

   1 ;RECS FOR INVENTORY
   2 CREATE,INF,0.001,,1,1;
   3 ACTIVITY;
   4 ASSIGN,{{IDXREC,1}},1;
   5 ACTIVITY;
   6 RECS_READ_1:
READ,CASER,YES,IDX,,{NSN,QTY,DATE,DELTIME,SHIPTIME,IPG,DUPRICE,DC1,F1,DC2,F2,DC3,F3},1
;
   7 ACTIVITY,,,IDX > 0;
   8 ACTIVITY,,,IDX == 0,"RECS_TERMINATE_1";
   9 RECS_GOON_1: GOON,1;
  10 ACTIVITY,,DATE - TNOW,DATE == IDXREC;
  11 ACTIVITY,,1,DATE > IDXREC,"RECS_ASSIGN_2";
  12 RECS_ASSIGN_1: ASSIGN,{{TRR,TRR + 1}},2;
  13 ACTIVITY,,,,"RECS_READ_1";
  14 ACTIVITY;
  15 GOON,1;
  16 ACTIVITY,,,IPG == 1;
  17 ACTIVITY,,,IPG == 2,"RECS4_ASSIGN_3";
  18 ACTIVITY,,,IPG == 3,"RECS4_ASSIGN_4";
  19 ASSIGN,{{TRR1,TRR1 + 1}},1;
  20 ACTIVITY;
  21 RECS4_GOON_2: GOON,1;
  22 ACTIVITY;
  23 ASSIGN,{{SZ[1],STRIB[1]}},1;
  24 ACTIVITY;
  25 FINDAR,1,STRIB[1] == SZ[1],1,FORWARD,1,,,IDX3,1;
  26 ACTIVITY,,,IDX3 >0;
  27 ACTIVITY,,,IDX3 == 0,"ERROR";
  28 GOON,1;
  29 ACTIVITY,,,,"Demand";
  30 ERROR: EVENT,3,1;
  31 ACTIVITY;
  32 ERROR_No_Inv_Rec: TERMINATE,1;
  33 RECS4_ASSIGN_3: ASSIGN,{{TRR2,TRR2 + 1}},1;
  34 ACTIVITY,,,,"RECS4_GOON_2";
  35 RECS4_ASSIGN_4: ASSIGN,{{TRR3,TRR3 + 1}},1;
  36 ACTIVITY,,,,"RECS4_GOON_2";
  37 RECS_ASSIGN_2: ASSIGN,{{IDXREC,IDXREC + 1}},1;
  38 ACTIVITY,,,,"RECS_GOON_1";
  39 RECS_TERMINATE_1: TERMINATE,INF;

RECS1ST - Pass 3 successfully read

Reading network REVIEWRQ - Pass 3...

   1 ;INVENTORY REVIEW (r, Q)
   2 ;Individual Inventory Item is Reviewed
   3 ;INV Entity incoming
   4 ;DEMAND ENTITY
   5 PROCBO: GOON,1;
   6 ACTIVITY,,0.0001;
   7
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL},{XX[16],DCF},{BONOSTK,BONOSTK - 1}},1;
   8 ACTIVITY;
```

```
   9 FINDAR,1,NSN == SZ[1] && DC == XX[15],1,FORWARD,1,,{{STOCK,STOCK -
XX[1]},{BODAYS,BODAYS+(TNOW-BOSTART)}},IDX,1;
  10 ACTIVITY,,,,"HAVESTOCK";
  11 REVIEW5_ASSIGN_2: ASSIGN,{{STOCK,STOCK - TQTY}},1;
  12 ACTIVITY;
  13 REVIEW5_ASSIGN_1:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
}},1;
  14 ACTIVITY;
  15 REVIEW5_FINDAR_1: FINDAR,2,NSN == SZ[1] && QTY <= XX[1] && DCSEL ==
XX[15],1,FORWARD,1,"PROCBO",{{TQTY,QTY}},IDX,1;
  16 ACTIVITY,,0.0001,IDX > 0,"REVIEW5_ASSIGN_2";
  17 ACTIVITY,,,IDX == 0,"REVIEW5_TERMINATE_1";
  18 REVIEW5_TERMINATE_1: TERMINATE,INF;
  19 REVIEWRQ_ASSIGN_ERROR: ASSIGN,{{TREVIEWERROR,TSTOCK*MAX(-0.2, MIN(0.2,
RNORM(0,1,1)))},{TREVIEWLEVEL,MAX(CEIL(TINVPOS+TREVIEWERROR),0)}},1;
  20 ACTIVITY;
  21 REVIEW4_ASSIGN_8:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
},{STOCK,TSTOCK},{INVPOS,TINVPOS}},1;
  22 ACTIVITY,,,,"REVIEW4_FINDAR_2";
  23 REVIEW4_FINDAR_2: FINDAR,1,STRIB[1]==SZ[1] && DC == XX[15] && Q > 0 &&
TREVIEWLEVEL <= R,1,FORWARD,1,,{{ORDERS,ORDERS+1}},IDX,1;
  24 ACTIVITY,,,IDX > 0,"ORDER";
  25 ACTIVITY,,,IDX == 0,"REVIEW_TERMINATE_1";
  26 ORDER: GOON,1;
  27 ACTIVITY,,,,"REVIEW4_ASSIGN_4";
  28 REVIEW4_ASSIGN_4:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
}},1;
  29 ACTIVITY,,,,"REVIEW4_FINDAR_1";
  30 REVIEW4_FINDAR_1: FINDAR,1,STRIB[1]==SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{TLTIME,LEADTIME},{INVPOS,INVPOS +
Q},{ORDERQTYSUM,ORDERQTYSUM+Q}},IDX,2;
  31 ACTIVITY,,RLOGN(TLTIME,0.1,1),IDX>0,"REVIEW4_ASSIGN_2";
  32 ACTIVITY,,,IDX > 0,"REVIEW4_COLCT_1";
  33 ACTIVITY,,,IDX == 0,"ERROR_REVIEW";
  34 REVIEW4_ASSIGN_2:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
}},1;
  35 ACTIVITY,,,,"UPDATE_INV";
  36 UPDATE_INV: ASSIGN,{{TIVCD,TIVCD + UPRICE*Q*DCD},{TIV,TIV + UPRICE*Q}},1;
  37 ACTIVITY,,,DC == 1,"REVIEW4_ASSIGN_1";
  38 ACTIVITY,,,DC == 2,"NODE_9";
  39 ACTIVITY,,,DC == 3,"LDINV2_ASSIGN_8";
  40 REVIEW4_ASSIGN_1: ASSIGN,{{TIDC1,TIDC1 + (Q*UPRICE)}},1;
  41 ACTIVITY;
  42 REVIEW3_FINDAR_2: FINDAR,1,STRIB[1] == SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{STOCK,STOCK + Q},{TSTOCK,STOCK},{TINVPOS,INVPOS}},IDX,1;
  43 ACTIVITY,,,IDX > 0;
  44 ACTIVITY,,,IDX == 0,"REVIEW4_EVENT_1";
  45 ASSIGN,{{STOCK,TSTOCK},{INVPOS,TINVPOS}},1;
  46 ACTIVITY;
  47
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
}},1;
  48 ACTIVITY,,,,"REVIEW5_FINDAR_1";
  49 REVIEW4_EVENT_1: EVENT,5,1;
  50 ACTIVITY;
  51 TERMINATE,1;
  52 NODE_9: ASSIGN,{{TIDC2,TIDC2 + (Q*UPRICE)}},1;
  53 ACTIVITY,,,,"REVIEW3_FINDAR_2";
  54 LDINV2_ASSIGN_8: ASSIGN,{{TIDC3,TIDC3 + (Q*UPRICE)}},1;
  55 ACTIVITY,,,,"REVIEW3_FINDAR_2";
  56 REVIEW4_COLCT_1: COLCT,19,TLTIME,"Lead Time",,,1;
  57 ACTIVITY;
  58 TERMINATE,INF;
  59 ERROR_REVIEW: TERMINATE,1;
  60 REVIEW_TERMINATE_1: TERMINATE,INF;
```

```
  61 COUNT_REVIEWS: FINDAR,1,STRIB[1]==SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{REVIEWS,REVIEWS+1},{STOCKLEVELSUM,STOCKLEVELSUM+STOCK},{TINVPOS,
INVPOS},{TSTOCK,STOCK}},IDX,1;
  62 ACTIVITY,,,ERRORRATE>0 && TSTOCK>20,"REVIEWRQ_ASSIGN_ERROR";
  63 ACTIVITY,,,ERRORRATE==0 || TSTOCK <= 20;
  64 REVIEWRQ_ASSIGN_NOERROR: ASSIGN,{{TREVIEWLEVEL,TINVPOS}},1;
  65 ACTIVITY,,,,"REVIEW4_ASSIGN_8";
  66 COUNT_REVIEWS_ASG: ASSIGN,{{SZ[1],STRIB[1]},{XX[15],DC}},1;
  67 ACTIVITY,,,,"COUNT_REVIEWS";
  68 REVIEWRQ: GOON,2;
  69 ACTIVITY,,REVIEWPERIOD,,"REVIEWRQ";
  70 ACTIVITY,,,,"COUNT_REVIEWS_ASG";

REVIEWRQ - Pass 3 successfully read

Reading network REVIEWSS - Pass 3...

   1 ;INVENTORY REVIEW (s,S)
   2 ;INV Entity incoming
   3 ;Individual Inventory Item is Reviewed
   4 ;DEMAND ENTITY
   5 SSPROCBO: GOON,1;
   6 ACTIVITY,,0.0001;
   7
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL},{XX[16],DCF},{BONOSTK,BONOSTK - 1}},1;
   8 ACTIVITY;
   9 FINDAR,1,NSN == SZ[1] && DC == XX[15],1,FORWARD,1,,{{STOCK,STOCK -
XX[1]},{BODAYS,BODAYS+(TNOW-BOSTART)}},IDX,2;
  10 ACTIVITY,,,,"HAVESTOCK";
  11 REVIEWSS_ASSIGN_ERROR: ASSIGN,{{TREVIEWERROR,TSTOCK*MAX(-0.2, MIN(0.2,
RNORM(0,1,1)))},{TREVIEWLEVEL,MAX(CEIL(INVPOS+TREVIEWERROR),0)}},1;
  12 ACTIVITY;
  13 REVIEWSS_ASSIGN_1:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],MAX(ATRIB[2],ATRIB[3
])},{XX[15],DC},{STOCK,TSTOCK},{INVPOS,TINVPOS}},1;
  14 ACTIVITY,,,,"REVIEWSS_FINDAR_1";
  15 REVIEWSS_FINDAR_1: FINDAR,1,STRIB[1]==SZ[1] && DC == XX[15] && Q > 0 &&
TREVIEWLEVEL <= R,1,FORWARD,1,,{{ORDERS,ORDERS+1}},IDX,1;
  16 ACTIVITY,,,IDX > 0,"ORDERSS";
  17 ACTIVITY,,,IDX == 0,"REVIEWSS_TERMINATE_1";
  18 ORDERSS: GOON,1;
  19 ACTIVITY,,,,"REVIEWSS_ASSIGN_2";
  20 REVIEWSS_ASSIGN_2:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
},{LASTORDERQ,MAX(Q-INVPOS,0)},{TQTY,LASTORDERQ}},1;
  21 ACTIVITY;
  22 REVIEWSS_FINDAR_2: FINDAR,1,STRIB[1]==SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{TLTIME,LEADTIME},{INVPOS,Q},{ORDERQTYSUM,ORDERQTYSUM+TQTY}},IDX,
2;
  23 ACTIVITY,,RLOGN(TLTIME,0.1,1),IDX>0,"REVIEWSS_ASSIGN_3";
  24 ACTIVITY,,,IDX > 0,"REVIEWSS_COLCT_1";
  25 ACTIVITY,,,IDX == 0,"SSERROR_REVIEW";
  26 REVIEWSS_ASSIGN_3:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
}},1;
  27 ACTIVITY,,,,"SSUPDATE_INV";
  28 SSUPDATE_INV: ASSIGN,{{TIVCD,TIVCD + UPRICE*LASTORDERQ*DCD},{TIV,TIV +
UPRICE*LASTORDERQ},{TQTY,LASTORDERQ}},1;
  29 ACTIVITY,,,DC == 1,"REVIEWSS_ASSIGN_4";
  30 ACTIVITY,,,DC == 2,"REVIEWSS_ASSIGN_5";
  31 ACTIVITY,,,DC == 3,"REVIEWSS_ASSIGN_6";
  32 REVIEWSS_ASSIGN_4: ASSIGN,{{TIDC1,TIDC1 + (LASTORDERQ*UPRICE)}},1;
  33 ACTIVITY;
  34 REVIEWSS_FINDAR_3: FINDAR,1,STRIB[1] == SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{STOCK,STOCK + TQTY},{TSTOCK,STOCK},{TINVPOS,INVPOS}},IDX,1;
  35 ACTIVITY,,,IDX > 0;
  36 ACTIVITY,,,IDX == 0,"REVIEWSS_EVENT_5";
  37 ASSIGN,{{STOCK,TSTOCK},{INVPOS,TINVPOS}},1;
  38 ACTIVITY;
```

```
  39
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
}},1;
  40 ACTIVITY;
  41 REVIEWSS_FINDAR_4: FINDAR,2,NSN == SZ[1] && QTY <= XX[1] && DCSEL ==
XX[15],1,FORWARD,1,"SSPROCBO",{{TQTY,QTY}},IDX,1;
  42 ACTIVITY,,0.0001,IDX > 0;
  43 ACTIVITY,,,IDX == 0,"REVIEWSS_TERMINATE_2";
  44 REVIEWSS_ASSIGN_7: ASSIGN,{{STOCK,STOCK - TQTY}},1;
  45 ACTIVITY;
  46 REVIEWSS_ASSIGN_8:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
}},1;
  47 ACTIVITY,,,,"REVIEWSS_FINDAR_4";
  48 REVIEWSS_TERMINATE_2: TERMINATE,INF;
  49 REVIEWSS_EVENT_5: EVENT,5,1;
  50 ACTIVITY;
  51 TERMINATE,1;
  52 REVIEWSS_ASSIGN_5: ASSIGN,{{TIDC2,TIDC2 + (LASTORDERQ*UPRICE)}},1;
  53 ACTIVITY,,,,"REVIEWSS_FINDAR_3";
  54 REVIEWSS_ASSIGN_6: ASSIGN,{{TIDC3,TIDC3 + (LASTORDERQ*UPRICE)}},1;
  55 ACTIVITY,,,,"REVIEWSS_FINDAR_3";
  56 REVIEWSS_COLCT_1: COLCT,19,TLTIME,"Lead Time",,,,1;
  57 ACTIVITY;
  58 TERMINATE,INF;
  59 SSERROR_REVIEW: TERMINATE,1;
  60 REVIEWSS_TERMINATE_1: TERMINATE,INF;
  61 SS_COUNT_REVIEWS: FINDAR,1,STRIB[1]==SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{REVIEWS,REVIEWS+1},{STOCKLEVELSUM,STOCKLEVELSUM+STOCK},{TINVPOS,
INVPOS},{TSTOCK,STOCK}},IDX,1;
  62 ACTIVITY,,,ERRORRATE>0 && TSTOCK > 20,"REVIEWSS_ASSIGN_ERROR";
  63 ACTIVITY,,,ERRORRATE==0 || TSTOCK <= 20;
  64 REVIEWSS_ASSIGN_NOERROR: ASSIGN,{{TREVIEWLEVEL,TINVPOS}},1;
  65 ACTIVITY,,,,"REVIEWSS_ASSIGN_1";
  66 SS_COUNT_REVIEWS_ASG: ASSIGN,{{SZ[1],STRIB[1]},{XX[15],DC}},1;
  67 ACTIVITY,,,,"SS_COUNT_REVIEWS";
  68 REVIEWSS: GOON,2;
  69 ACTIVITY,,REVIEWPERIOD,,"REVIEWSS";
  70 ACTIVITY,,,,"SS_COUNT_REVIEWS_ASG";

REVIEWSS - Pass 3 successfully read

Reading network STATS1ST - Pass 3...

   1 ;STAT COLLECTION
   2 CREATE,30,30.0045,,INF,11;
   3 ACTIVITY;
   4 ACTIVITY,,,,"STATS_COLCT_1";
   5 ACTIVITY,,,,"STATS_COLCT_3";
   6 ACTIVITY,,,,"STATS_COLCT_4";
   7 ACTIVITY,,,,"STATS_COLCT_6";
   8 ACTIVITY,,,,"STATS_COLCT_7";
   9 ACTIVITY,,,,"STATS_COLCT_8";
  10 ACTIVITY,,,,"STATS2_COLCT_1";
  11 ACTIVITY,,,,"STATS_GOON_1";
  12 ACTIVITY,,,,"STATS5_ASSIGN_2";
  13 ACTIVITY,,,,"STATS_WRITE_1";
  14 COLCT,3,TRR,"TRR",,,,1;
  15 ACTIVITY;
  16 STATS_TERMINATE_1: TERMINATE,INF;
  17 STATS_COLCT_1: COLCT,2,TRS,"TRS",,,,1;
  18 ACTIVITY,,,,"STATS_TERMINATE_1";
  19 STATS_COLCT_3: COLCT,4,BO,"BO",,,,1;
  20 ACTIVITY,,,,"STATS_TERMINATE_1";
  21 STATS_COLCT_4: COLCT,5,BOPM,"BOPM",,,,1;
  22 ACTIVITY,,,,"STATS_TERMINATE_1";
  23 STATS_COLCT_6: COLCT,9,VMITOT,"TOTAL VMI",,,,1;
  24 ACTIVITY,,,,"STATS_TERMINATE_1";
  25 STATS_COLCT_7: COLCT,10,TRNPERMON,"TRANPERMON",,,,1;
  26 ACTIVITY,,,,"STATS_TERMINATE_1";
  27 STATS_COLCT_8: COLCT,11,NNQ(2),"BO_TRAN",,,,1;
```

```
  28 ACTIVITY,,,,"STATS_TERMINATE_1";
  29 STATS2_COLCT_1: COLCT,13,CDTRNPMON,"CDTRANPM",,,,1;
  30 ACTIVITY,,,,"STATS_TERMINATE_1";
  31 STATS_GOON_1: GOON,1;
  32 ACTIVITY,,,TRR + BOPM > 0;
  33 ACTIVITY,,,,"STATS_COLCT_5";
  34 STATS_COLCT_2: COLCT,1,TRS/(TRR+BOPM),"FR",,,,1;
  35 ACTIVITY;
  36 STATS_COLCT_5: COLCT,6,XX[6],"Mon Rec Val";
  37 ACTIVITY;
  38 ACTIVITY,,,,"STATS2_COLCT_3";
  39 ACTIVITY,,,,"STATS2_COLCT_4";
  40 ACTIVITY,,,,"STATS2_COLCT_5";
  41 COLCT,14,MFCDC1,"MonShipDC1",,,,1;
  42 ACTIVITY;
  43 STATS2_COLCT_2: COLCT,12,TRVCD,"CD_MonRecVal",,,,1;
  44 ACTIVITY,,,TRR>0;
  45 ACTIVITY,,0.02,,"STATS_ASSIGN_1";
  46 ASSIGN,{{POSOT,1-BO/TRR}},1;
  47 ACTIVITY;
  48 COLCT,8,POSOT,"Perc Ord Ship OnTime",,,,1;
  49 ACTIVITY,,.02;
  50 STATS_ASSIGN_1:
ASSIGN,{{TRR,0},{TRS,0},{BOPM,BO},{BO,0},{TRR1,0},{TRR2,0},{TRR3,0},{TRS1,0},{TRS2,0},
{TRS3,0},{DLNI,0},{DLDT,0},{DLB,0},{GBL,0}},1;
  51 ACTIVITY;
  52
ASSIGN,{{XX[6],0},{TRNPERMON,0},{TRVCD,0},{CDTRNPMON,0},{MFCDC3,0},{MFCDC2,0},{MFCDC1,
0},{NSEL1,0},{NSEL2,0},{NSEL3,0}},1;
  53 ACTIVITY;
  54 TERMINATE,INF;
  55 STATS2_COLCT_3: COLCT,15,MFCDC2,"MonShipDC2",,,,1;
  56 ACTIVITY,,,,"STATS2_COLCT_2";
  57 STATS2_COLCT_4: COLCT,16,MFCDC3,"MonShipDC3",,,,1;
  58 ACTIVITY,,,,"STATS2_COLCT_2";
  59 STATS2_COLCT_5: COLCT,17,MFCDC3 + MFCDC2 + MFCDC1,"MonShipTot",,,,1;
  60 ACTIVITY,,,,"STATS2_COLCT_2";
  61 STATS5_ASSIGN_2: ASSIGN,{{IDX2,1}},1;
  62 ACTIVITY;
  63 GETINVVAL:
FINDAR,1,LTRIB[4]==IDX2,1,FORWARD,1,,{{SZ[1],STRIB[1]},{NREVIEWS,REVIEWS*0.25},{REVIEW
S,0},{NORDERS,ORDERS*0.5},{ORDERS,0},{NBACKORDERS,BACKORDERS*UPRICE*4},{BACKORDERS,0},
{NBODAYS,MIN(BODAYS/7.0,8)},{NSPORDERS,SPORDERS*40},{BODAYS,0},{SPORDERS,0},{TAVGSTOCK
LVL,NINT((STOCKLEVELSUM*REVIEWPERIOD)/30)},{TAVGINVVALUE,TAVGSTOCKLVL*UPRICE},{STOCKLE
VELSUM,0},{TORDERQTYVALUE,ORDERQTYSUM*UPRICE},{ORDERQTYSUM,0}},IDX,1;
  64 ACTIVITY,,,IDX>0;
  65 ACTIVITY,,,IDX==0,"STATS5_TERMINATE_1";
  66
ASSIGN,{{COSTOUT,STRCAT(SZ[1],'.CST')},{IDX2,IDX2+1},{MCOST,NSPORDERS+(EXP(NBODAYS)-
1)*240+NBACKORDERS+(XX[28]*0.005)+NREVIEWS+NORDERS+(XX[28]*0.01)}},1;
  67 ACTIVITY;
  68 WRITECOSTS: EVENT,10,2;
  69 ACTIVITY,,,,"GETINVVAL";
  70 ACTIVITY;
  71 STATS5_TERMINATE_1: TERMINATE,INF;
  72 STATS_WRITE_1:
WRITE,CASEOUT,YES,"%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%
f,%f,%f,%f,%f,%f,%f,%f,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%f,%f,%f,%f,%f,%f,%d,%d,%d,%d\n",
{TNOW,CCAVG(1),CCAVG(2),CCAVG(3),CCAVG(4),CCAVG(5),CCAVG(6),CCAVG(7),CCAVG(8),CCAVG(9)
,CCAVG(10),CCAVG(11),CCAVG(12),CCAVG(13),TTAVG(1),TTAVG(2),TTAVG(3),TTAVG(4),0.0,0.0,0
.0,CCAVG(14),CCAVG(15),CCAVG(16),CCAVG(17),TTAVG(5),TTAVG(6),TTAVG(7),CCAVG(18),CCAVG(
19),BONOSTK,TRR1,TRR2,TRR3,TRS1,TRS2,TRS3,DLNI,DLDT,DLB,CCAVG(20),CCAVG(21),CCAVG(22),
CCAVG(23),CCAVG(24),CCAVG(25),GBL,NSEL1,NSEL2,NSEL3},1;
  73 ACTIVITY;
  74 TERMINATE,INF;
  75 ;Order value + (Special Order Cost)+(Idle Worker Cost)+(Initial Backorder
Cost)+(Holding Cost)+(Review Cost)+(Order Placement Cost)+(Cost of Capital)

STATS1ST - Pass 3 successfully read
```

```
Translated network file INIT.TRN successfully written
```

```
AweSim Input Translator, version 3.0
Copyright (C) 1999 Symix Systems, Inc.

Reading control CONPLUS ...

   1 ARRAY,1,3,{2,5,10};
   2 GEN,,,,1,YES,YES;
   3 LIMITS,38,37,5,16,11,1;
   4 PRIORITY,{{1,FIFO},{2,LVF(STRIB[1])}};
   5
EQUIVALENCE,{{NSN,STRIB[1]},{STOCK,ATRIB[1]},{Q,ATRIB[2]},{R,ATRIB[3]},{IDX,LL[4]},{UP
RICE,ATRIB[4]},{LEADTIME,ATRIB[5]},{LOC,LTRIB[1]},{CURDAY,XX[10]},{IDXREC,LL[6]},{INVP
OS,ATRIB[8]},{CD,LTRIB[2]},{DC,LTRIB[3]},{TIDC1,XX[17]},{TIDC2,XX[18]},{TIDC3,XX[19]},
{RECNUM,LTRIB[4]},{ORDERS,LTRIB[5]},{REVIEWS,LTRIB[6]},{BACKORDERS,ATRIB[14]},{BOSTART
,ATRIB[9]},{BODAYS,ATRIB[10]},{SPORDERS,LTRIB[7]},{REVIEWPERIOD,LTRIB[8]},{LASTORDERQ,
ATRIB[11]},{POLICY,LTRIB[10]},{STOCKLEVELSUM,ATRIB[12]},{ORDERQTYSUM,ATRIB[13]},{DDEST
,ATRIB[14]},{SDEST,ATRIB[15]},{DEMANDTYPE,LTRIB[11]},{ERRORRATE,ATRIB[16]}};
   6
EQUIVALENCE,{{QTY,ATRIB[1]},{DATE,ATRIB[2]},{SHIPTIME,ATRIB[3]},{DELTIME,ATRIB[4]},{IP
G,ATRIB[5]},{DUPRICE,ATRIB[6]},{STIMESAM,ATRIB[7]},{BOFLAG,LTRIB[1]},{TQ,ATRIB[8]},{DL
OC,LTRIB[2]},{DCD,LTRIB[3]},{DC1,LTRIB[4]},{F1,ATRIB[9]},{DC2,LTRIB[5]},{F2,ATRIB[10]}
,{DC3,LTRIB[6]},{F3,ATRIB[11]},{DCSEL,LTRIB[7]},{DCF,ATRIB[12]},{DLEADTIME,ATRIB[13]}}
;
   7
EQUIVALENCE,{{TRR,LL[1]},{TRS,LL[2]},{BO,LL[3]},{TIV,XX[5]},{BOPM,LL[5]},{POSOT,XX[8]}
,{COC,XX[9]},{TLTIME,XX[11]},{BONOSTK,LL[7]},{TEMPQ,XX[12]},{IDX2,LL[9]},{TEMPLOC,LL[1
0]},{VMITOT,LL[11]},{TRNPERMON,LL[12]},{TIVCD,XX[13]},{TRVCD,XX[14]},{TEMPCD,LL[13]},{
CDTRNPMON,LL[14]},{NUMDC,LL[15]},{MFCDC1,XX[20]},{MFCDC2,XX[21]},{MFCDC3,XX[22]},{TRSP
M,LL[16]},{STMON,LL[17]},{TRSHOLD,LL[18]},{IDX3,LL[19]},{TRR1,LL[20]},{TRR2,LL[21]},{T
RR3,LL[22]},{TRS1,LL[23]},{TRS2,LL[24]},{TRS3,LL[25]},{DLNI,LL[26]},{DLDT,LL[27]},{DLB
,LL[28]},{GBL,LL[29]},{NSEL1,LL[30]},{NSEL2,LL[31]},{NSEL3,LL[32]},{CASER,SZ[2]},{CASE
INV,SZ[3]},{CASEOUT,SZ[4]},{TSTOCK,XX[23]},{TINVPOS,XX[24]},{TQTY,XX[25]},{DIRECTION,L
L[35]},{COSTOUT,SZ[5]},{NREVIEWS,XX[30]},{NORDERS,XX[29]},{NBACKORDERS,XX[37]},{NBODAY
S,XX[31]},{NSPORDERS,LL[37]},{DELTA1,XX[32]},{DELTA2,XX[33]},{DELTA3,XX[34]},{TAVGSTOC
KLVL,LL[34]},{TAVGINVVALUE,XX[28]},{TORDERQTYVALUE,XX[36]},{TREVIEWERROR,XX[38]},{TREV
IEWLEVEL,LL[36]},{MCOST,XX[35]}};
   8 INTLC,{{COC,0.12},{NUMDC,1}};
   9 INITIALIZE,0.0,1096,YES,,NO;
  10 TIMST,1,TIV,"Total Inventory per Month",0,0.0,1.0;
  11 TIMST,4,TIVCD,"CostDriver Inventory per Month",0,0.0,1.0;
  12 TIMST,3,BO,"BO per month",0,0.0,1.0;
  13 TIMST,2,COC*TIV,"Cost of Cap",0,0.0,1.0;
  14 TIMST,5,TIDC1,"TotInvDC1",0,0.0,1.0;
  15 TIMST,6,TIDC2,"TotInvDC2",0,0.0,1.0;
  16 TIMST,7,TIDC3,"TotInvDC3",0,0.0,1.0;
  17 MONTR,SUMMARY,30.005,30;
  18 MONTR,CLEAR,30.01,30;
  19 NET;
  20
INTLC,{{CASER,"DEMAND.TXT"},{CASEINV,"INVENTORY2.TXT"},{CASEOUT,"COSTOUT_INC.TXT"},{NU
MDC,1},{DIRECTION,1}};
  21 FIN;

CONPLUS successfully read


Translated file PLUS successfully written
Reading network DEMAND5 - Pass 1...


DEMAND5 - Pass 1 successfully read

Reading network LDINVSP - Pass 1...


LDINVSP - Pass 1 successfully read

Reading network RECS5 - Pass 1...
```

```
RECS5 - Pass 1 successfully read

Reading network REVIEWRQ - Pass 1...

REVIEWRQ - Pass 1 successfully read

Reading network REVIEWSS - Pass 1...

REVIEWSS - Pass 1 successfully read

Reading network STATS5 - Pass 1...

STATS5 - Pass 1 successfully read

Reading network DEMAND5 - Pass 2...

DEMAND5 - Pass 2 successfully read

Reading network LDINVSP - Pass 2...

LDINVSP - Pass 2 successfully read

Reading network RECS5 - Pass 2...

RECS5 - Pass 2 successfully read

Reading network REVIEWRQ - Pass 2...

REVIEWRQ - Pass 2 successfully read

Reading network REVIEWSS - Pass 2...

REVIEWSS - Pass 2 successfully read

Reading network STATS5 - Pass 2...

STATS5 - Pass 2 successfully read

Reading network DEMAND5 - Pass 3...

   1 ;DEMAND FOR INVENTORY
   2 DEMAND: GOON,1;
   3 ACTIVITY;
   4 Samp_Shiptime: ASSIGN,{{DELTIME,DELTIME*0.9},{STIMESAM,SHIPTIME*0.9}},1;
   5 ACTIVITY,,,,"SELDC";
   6 DEMAND2:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL},{XX[16],DCF}},1;
   7 ACTIVITY;
   8 FINDAR,1,NSN == SZ[1] && STOCK >= XX[1] && DC ==
XX[15],1,FORWARD,1,,{{STOCK,STOCK - XX[1]},{INVPOS,INVPOS - XX[1]},{TEMPCD,CD}},IDX,1;
   9 ACTIVITY,,,IDX == 0,"BKORD";
  10 ACTIVITY,,,IDX > 0,"DEMAND5_ASSIGN_2";
  11 BKORD:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL},{XX[16],DCF}},1;
  12 ACTIVITY;
  13 CHECK: FINDAR,1,NSN == SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{TEMPLOC,LOC},{TLTIME,LEADTIME},{INVPOS,INVPOS -
XX[1]},{TEMPCD,CD},{BACKORDERS,BACKORDERS+XX[1]},{BOSTART,TNOW}},IDX,1;
  14 ACTIVITY;
```

```
  15 DEMAND_ASSIGN_3:
ASSIGN,{{DLOC,TEMPLOC},{DLEADTIME,TLTIME},{DCD,TEMPCD},{BOFLAG,1},{BONOSTK,BONOSTK +
1}},1;
  16 ACTIVITY,,,,"DEMAND5_ASSIGN_1";
  17 DEMAND5_ASSIGN_1:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL}},1;
  18 ACTIVITY;
  19 DEMAND5_FINDAR_1: FINDAR,1,NSN == SZ[1] && DC == XX[15] && Q ==
0,1,FORWARD,1,,{{SPORDERS,SPORDERS+1}},IDX,1;
  20 ACTIVITY,,,IDX > 0,"SPORD";
  21 ACTIVITY,,,IDX == 0;
  22 BORDER: EVENT,2,1;
  23 ACTIVITY;
  24 TERMINATE,INF;
  25 DEMAND5_ASSIGN_2: ASSIGN,{{DCD,TEMPCD}},2;
  26 ACTIVITY;
  27 HAVESTOCK:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL},{XX[16],DCF}},1;
  28 ACTIVITY;
  29 DEMAND4_FINDAR_1: FINDAR,1,NSN == SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{TEMPLOC,LOC},{TLTIME,LEADTIME}},IDX,2;
  30 ACTIVITY;
  31 DEMAND4_ASSIGN_5: ASSIGN,{{DLOC,TEMPLOC},{DLEADTIME,TLTIME}},1;
  32 ACTIVITY,,STIMESAM,,"DEMAND_ASSIGN_1";
  33 DEMAND_ASSIGN_1: ASSIGN,{{TIV,TIV - (QTY*DUPRICE)},{TRVCD,TRVCD +
QTY*DUPRICE*DCD},{XX[6],XX[6] + QTY*DUPRICE},{TIVCD,TIVCD - QTY*DUPRICE*DCD}},1;
  34 ACTIVITY,,,DCSEL== 1;
  35 ACTIVITY,,,DCSEL == 2,"LDINV2_ASSIGN_1";
  36 ACTIVITY,,,DCSEL == 3,"LDINV2_ASSIGN_2";
  37 ASSIGN,{{TIDC1,TIDC1 - (QTY*DUPRICE)},{MFCDC1,MFCDC1 + DCF}},1;
  38 ACTIVITY;
  39 DEMAND_COLCT_1: COLCT,7,TNOW-DATE,"Time To Ship",5,1,1,1;
  40 ACTIVITY,,,IPG==1;
  41 ACTIVITY,,,IPG==2,"DEMAND4_COLCT_5";
  42 ACTIVITY,,,IPG==3,"DEMAND4_COLCT_6";
  43 COLCT,20,TNOW - DATE,"TTS_IPG1",,,,1;
  44 ACTIVITY,,DELTIME;
  45 DEMAND4_COLCT_4: COLCT,18,TNOW - DATE,"Time To Deliver",5,1,1,1;
  46 ACTIVITY,,,IPG==1;
  47 ACTIVITY,,,IPG == 2,"DEMAND4_COLCT_2";
  48 ACTIVITY,,,IPG == 3,"DEMAND4_COLCT_3";
  49 COLCT,23,TNOW - DATE,"TTD-IPG1",,,,1;
  50 ACTIVITY;
  51 DEMAND4_GOON_1: GOON,1;
  52 ACTIVITY,,,TNOW - DATE <= ARRAY[1,IPG];
  53 ACTIVITY,,,TNOW - DATE > ARRAY[1,IPG],"DEMAND_ASSIGN_2";
  54 ACTIVITY,,,,"DEMAND4_EVENT_1";
  55 SHIPPED: ASSIGN,{{TRS,TRS + 1}},1;
  56 ACTIVITY,,,IPG == 1;
  57 ACTIVITY,,,IPG == 2,"DEMAND4_ASSIGN_1";
  58 ACTIVITY,,,IPG == 3,"DEMAND4_ASSIGN_2";
  59 ASSIGN,{{TRS1,TRS1 + 1}},1;
  60 ACTIVITY;
  61 DEMAND4_TERMINATE_1: TERMINATE,INF;
  62 DEMAND4_ASSIGN_1: ASSIGN,{{TRS2,TRS2 + 1}},1;
  63 ACTIVITY,,,,"DEMAND4_TERMINATE_1";
  64 DEMAND4_ASSIGN_2: ASSIGN,{{TRS3,TRS3 + 1}},1;
  65 ACTIVITY,,,,"DEMAND4_TERMINATE_1";
  66 DEMAND_ASSIGN_2: ASSIGN,{{BO,BO + 1}},1;
  67 ACTIVITY;
  68 GOON,1;
  69 ACTIVITY,,,BOFLAG == 1 && ((DELTIME + STIMESAM) > ARRAY[1,IPG]);
  70 ACTIVITY,,,BOFLAG == 1,"DEMAND4_ASSIGN_3";
  71 ACTIVITY,,,((DELTIME + STIMESAM) > ARRAY[1,IPG]),"DEMAND4_ASSIGN_4";
  72 ACTIVITY,,,,"DEMAND4_EVENT_2";
  73 ASSIGN,{{DLB,DLB + 1}},1;
  74 ACTIVITY,,,,"DEMAND4_TERMINATE_1";
  75 DEMAND4_ASSIGN_3: ASSIGN,{{DLNI,DLNI + 1}},1;
  76 ACTIVITY,,,,"DEMAND4_TERMINATE_1";
```

```
 77 DEMAND4_ASSIGN_4: ASSIGN,{{DLDT,DLDT + 1}},1;
 78 ACTIVITY,,,,"DEMAND4_TERMINATE_1";
 79 DEMAND4_EVENT_2: EVENT,7,1;
 80 ACTIVITY;
 81 TERMINATE,1;
 82 DEMAND4_EVENT_1: EVENT,6,1;
 83 ACTIVITY;
 84 TERMINATE,1;
 85 DEMAND4_COLCT_2: COLCT,24,TNOW - DATE,"TTD-IPG2",,,,1;
 86 ACTIVITY,,,,"DEMAND4_GOON_1";
 87 DEMAND4_COLCT_3: COLCT,25,TNOW - DATE,"TTD-IPG3",,,,1;
 88 ACTIVITY,,,,"DEMAND4_GOON_1";
 89 DEMAND4_COLCT_5: COLCT,21,TNOW - DATE,"TTS_IPG2",,,,1;
 90 ACTIVITY,,DELTIME,,"DEMAND4_COLCT_4";
 91 DEMAND4_COLCT_6: COLCT,22,TNOW - DATE,"TTS_IPG3",,,,1;
 92 ACTIVITY,,DELTIME,,"DEMAND4_COLCT_4";
 93 LDINV2_ASSIGN_1: ASSIGN,{{TIDC2,TIDC2 - (QTY*DUPRICE)},{MFCDC2,MFCDC2 + DCF}},1;
 94 ACTIVITY,,,,"DEMAND_COLCT_1";
 95 LDINV2_ASSIGN_2: ASSIGN,{{TIDC3,TIDC3 - (QTY*DUPRICE)},{MFCDC3,MFCDC3 + DCF}},1;
 96 ACTIVITY,,,,"DEMAND_COLCT_1";
 97 ;CHECK FOR STOCK ETC UPDATE
 98 ;BO Order-Demand entity only
 99 SPORD: GOON,1;
100 ACTIVITY;
101
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL},{XX[16],DCF}},1;
102 ACTIVITY;
103 FINDAR,1,STRIB[1] == SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{TLTIME,LEADTIME},{ORDERQTYSUM,ORDERQTYSUM+XX[1]}},IDX,2;
104 ACTIVITY,,RLOGN(TLTIME,0.1,1),IDX > 0;
105 ACTIVITY,,,IDX > 0,"DEMAND4_COLCT_1";
106
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL},{XX[16],DCF}},1;
107 ACTIVITY;
108 FINDAR,1,NSN == SZ[1] && DC == XX[15],1,FORWARD,1,,{{BODAYS,BODAYS+(TNOW-
BOSTART)},{INVPOS,INVPOS+XX[1]}},IDX,1;
109 ACTIVITY;
110 ASSIGN,{{BONOSTK,BONOSTK - 1},{TRVCD,TRVCD + QTY*DUPRICE*DCD},{XX[6],XX[6] +
QTY*DUPRICE}},1;
111 ACTIVITY,,,DCSEL== 1;
112 ACTIVITY,,,DCSEL == 2,"DEMAND_ASSIGN_10";
113 ACTIVITY,,,DCSEL == 3,"DEMAND_ASSIGN_9";
114 ASSIGN,{{MFCDC1,MFCDC1 + DCF}},1;
115 ACTIVITY;
116 DEMAND_COLCT_2: GOON,1;
117 ACTIVITY,,STIMESAM,,"DEMAND_COLCT_1";
118 DEMAND_ASSIGN_10: ASSIGN,{{MFCDC2,MFCDC2 + DCF}},1;
119 ACTIVITY,,,,"DEMAND_COLCT_2";
120 DEMAND_ASSIGN_9: ASSIGN,{{MFCDC3,MFCDC3 + DCF}},1;
121 ACTIVITY,,,,"DEMAND_COLCT_2";
122 DEMAND4_COLCT_1: COLCT,19,TLTIME,"Lead Time",,,,1;
123 ACTIVITY;
124 TERMINATE,INF;
125 ;Select DC to use
126 SELDC:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
1},{XX[16],F1}},1;
127 ACTIVITY;
128 FINDAR,1,NSN == SZ[1] && STOCK >= XX[1] && DC == XX[15],1,FORWARD,1,,,IDX,1;
129 ACTIVITY,,,IDX > 0;
130 ACTIVITY,,,IDX == 0 && NUMDC > 1,"DEMAND_ASSIGN_7";
131 ACTIVITY,,,IDX == 0 && NUMDC == 1,"DEMAND_ASSIGN_8";
132 ASSIGN,{{DCSEL,DC1},{DCF,F1},{NSEL1,NSEL1 + 1}},1;
133 ACTIVITY,,,,"DEMAND2";
134 DEMAND_ASSIGN_7:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
2},{XX[16],F2}},1;
135 ACTIVITY;
136 FINDAR,1,NSN == SZ[1] && STOCK >= XX[1] && DC == XX[15],1,FORWARD,1,,,IDX,1;
```

```
 137 ACTIVITY,,,IDX > 0;
 138 ACTIVITY,,,IDX == 0 && NUMDC > 2,"DEMAND_ASSIGN_5";
 139 ACTIVITY,,,IDX == 0 && NUMDC == 2,"DEMAND_ASSIGN_6";
 140 ASSIGN,{{DCSEL,DC2},{DCF,F2},{NSEL2,NSEL2 + 1}},1;
 141 ACTIVITY,,,,"DEMAND2";
 142 DEMAND_ASSIGN_5:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
3},{XX[16],F3}},1;
 143 ACTIVITY;
 144 FINDAR,1,NSN == SZ[1] && STOCK >= XX[1] && DC == XX[15],1,FORWARD,1,,,IDX,1;
 145 ACTIVITY,,,IDX > 0;
 146 ACTIVITY,,,IDX == 0,"DEMAND_ASSIGN_4";
 147 ASSIGN,{{DCSEL,DC3},{DCF,F3},{NSEL3,NSEL3 + 1}},1;
 148 ACTIVITY,,,,"DEMAND2";
 149 DEMAND_ASSIGN_4: ASSIGN,{{DCSEL,DC1},{DCF,F1},{NSEL1,NSEL1 + 1}},1;
 150 ACTIVITY,,,,"DEMAND2";
 151 DEMAND_ASSIGN_6: ASSIGN,{{DCSEL,DC1},{DCF,F1},{NSEL1,NSEL1 + 1}},1;
 152 ACTIVITY,,,,"DEMAND2";
 153 DEMAND_ASSIGN_8: ASSIGN,{{DCSEL,DC1},{DCF,F1},{NSEL1,NSEL1+1}},1;
 154 ACTIVITY,,,,"DEMAND2";

DEMAND5 - Pass 3 successfully read

Reading network LDINVSP - Pass 3...

   1 ;Load Inventory Files
   2 CREATE,INF,0.0,,1,1;
   3 ACTIVITY;
   4 NET1_READ_1:
READ,CASEINV,NO,IDX,,{RECNUM,NSN,STOCK,Q,R,REVIEWPERIOD,POLICY,UPRICE,LEADTIME,LOC,CD,
DC,DDEST,SDEST,DEMANDTYPE,ERRORRATE},1;
   5 ACTIVITY,,,IDX > 0;
   6 ACTIVITY,,,IDX == 0,"NET1_TERMINATE_1";
   7 ReadParams: EVENT,8,2;
   8 ACTIVITY;
   9 ACTIVITY,,,IDX2==1,"LDINVSP_ASSIGN_1";
  10 ACTIVITY,,,IDX2==2,"LDINVSP_ASSIGN_2";
  11 ACTIVITY,,,IDX2==3,"LDINVSP_ASSIGN_3";
  12
ASSIGN,{{STOCK,MAX(STOCK,0)},{INVPOS,STOCK},{ORDERS,0},{REVIEWS,0},{BACKORDERS,0},{STO
CKLEVELSUM,0},{ORDERQTYSUM,0}},1;
  13 ACTIVITY;
  14 LDINVSP_EVENT_1: EVENT,1,1;
  15 ACTIVITY,,,LOC > 1;
  16 ACTIVITY,,,LOC == 1,"LDINV_ASSIGN_6";
  17 LDINV_ASSIGN_1: ASSIGN,{{TIV,TIV + MAX(UPRICE*STOCK,0)}},1;
  18 ACTIVITY,,,DC == 1;
  19 ACTIVITY,,,DC == 2,"LDINV2_ASSIGN_9";
  20 ACTIVITY,,,DC == 3,"LDINV2_ASSIGN_7";
  21 ACTIVITY,,,DC < 1 || DC > 3,"LDINV2_EVENT_1";
  22 ASSIGN,{{TIDC1,TIDC1 + MAX(UPRICE*STOCK,0)}},1;
  23 ACTIVITY;
  24 LDINV2_GOON_1: GOON,2;
  25 ACTIVITY;
  26 ACTIVITY,,,,"LDINVSP_GOON_1";
  27 GOON,1;
  28 ACTIVITY,,,CD == 0,"NET1_READ_1";
  29 ACTIVITY,,,CD == 1;
  30 LDINV_ASSIGN_3: ASSIGN,{{TIVCD,TIVCD + MAX(UPRICE*STOCK,0)}},1;
  31 ACTIVITY,,,,"NET1_READ_1";
  32 LDINVSP_GOON_1: GOON,1;
  33 ACTIVITY,,UNFRM(0,(REVIEWPERIOD-1)),POLICY <= 4,"REVIEWRQ";
  34 ACTIVITY,,UNFRM(0,(REVIEWPERIOD-1)),POLICY >= 5,"REVIEWSS";
  35 ACTIVITY;
  36 PolicyError: EVENT,9,1;
  37 ACTIVITY;
  38 TERMINATE,INF;
  39 LDINV2_ASSIGN_9: ASSIGN,{{TIDC2,TIDC2 + MAX(UPRICE*STOCK,0)}},1;
  40 ACTIVITY,,,,"LDINV2_GOON_1";
  41 LDINV2_ASSIGN_7: ASSIGN,{{TIDC3,TIDC3 + MAX(UPRICE*STOCK,0)}},1;
  42 ACTIVITY,,,,"LDINV2_GOON_1";
```

```
  43 LDINV2_EVENT_1: EVENT,4,1;
  44 ACTIVITY;
  45 ERROR_wrong_DC: TERMINATE,1;
  46 LDINV_ASSIGN_6: ASSIGN,{{VMITOT,VMITOT + 1}},1;
  47 ACTIVITY,,,,"LDINV_ASSIGN_1";
  48 LDINVSP_ASSIGN_1: ASSIGN,{{Q,MAX(NINT(Q+(DIRECTION*XX[32]*XX[26])),0)}},1;
  49 ACTIVITY,,,,"LDINVSP_EVENT_1";
  50 LDINVSP_ASSIGN_2:
ASSIGN,{{Q,MAX(NINT(Q+(DIRECTION*XX[32]*XX[26])),0)},{R,MAX(NINT(R+(DIRECTION*XX[33]*X
X[26])),0)}},1;
  51 ACTIVITY,,,,"LDINVSP_EVENT_1";
  52 LDINVSP_ASSIGN_3:
ASSIGN,{{Q,MAX(NINT(Q+(DIRECTION*XX[32]*XX[26])),0)},{R,MAX(NINT(R+(DIRECTION*XX[33]*X
X[26])),0)},{REVIEWPERIOD,MAX(NINT(REVIEWPERIOD+(DIRECTION*XX[34]*XX[26])),1)}},1;
  53 ACTIVITY,,,,"LDINVSP_EVENT_1";
  54 NET1_TERMINATE_1: TERMINATE,INF;

LDINVSP - Pass 3 successfully read

Reading network RECS5 - Pass 3...

   1 ;RECS FOR INVENTORY
   2 CREATE,INF,0.001,,1,1;
   3 ACTIVITY;
   4 ASSIGN,{{IDXREC,1}},1;
   5 ACTIVITY;
   6 RECS_READ_1:
READ,CASER,NO,IDX,,{NSN,QTY,DATE,DELTIME,SHIPTIME,IPG,DUPRICE,DC1,F1,DC2,F2,DC3,F3},1;
   7 ACTIVITY,,,IDX > 0;
   8 ACTIVITY,,,IDX == 0,"RECS_TERMINATE_1";
   9 RECS_GOON_1: GOON,1;
  10 ACTIVITY,,DATE - TNOW,DATE == IDXREC;
  11 ACTIVITY,,1,DATE > IDXREC,"RECS_ASSIGN_2";
  12 RECS_ASSIGN_1: ASSIGN,{{TRR,TRR + 1}},2;
  13 ACTIVITY,,,,"RECS_READ_1";
  14 ACTIVITY;
  15 GOON,1;
  16 ACTIVITY,,,IPG == 1;
  17 ACTIVITY,,,IPG == 2,"RECS4_ASSIGN_3";
  18 ACTIVITY,,,IPG == 3,"RECS4_ASSIGN_4";
  19 ASSIGN,{{TRR1,TRR1 + 1}},1;
  20 ACTIVITY;
  21 RECS4_GOON_2: GOON,1;
  22 ACTIVITY;
  23 ASSIGN,{{SZ[1],STRIB[1]}},1;
  24 ACTIVITY;
  25 FINDAR,1,STRIB[1] == SZ[1],1,FORWARD,1,,,IDX3,1;
  26 ACTIVITY,,,IDX3 >0;
  27 ACTIVITY,,,IDX3 == 0,"ERROR";
  28 GOON,1;
  29 ACTIVITY,,,,"Demand";
  30 ERROR: EVENT,3,1;
  31 ACTIVITY;
  32 ERROR_No_Inv_Rec: TERMINATE,1;
  33 RECS4_ASSIGN_3: ASSIGN,{{TRR2,TRR2 + 1}},1;
  34 ACTIVITY,,,,"RECS4_GOON_2";
  35 RECS4_ASSIGN_4: ASSIGN,{{TRR3,TRR3 + 1}},1;
  36 ACTIVITY,,,,"RECS4_GOON_2";
  37 RECS_ASSIGN_2: ASSIGN,{{IDXREC,IDXREC + 1}},1;
  38 ACTIVITY,,,,"RECS_GOON_1";
  39 RECS_TERMINATE_1: TERMINATE,INF;

RECS5 - Pass 3 successfully read

Reading network REVIEWRQ - Pass 3...

   1 ;INVENTORY REVIEW (r, Q)
   2 ;Individual Inventory Item is Reviewed
   3 ;INV Entity incoming
   4 ;DEMAND ENTITY
   5 PROCBO: GOON,1;
```

```
   6 ACTIVITY,,0.0001;
   7
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL},{XX[16],DCF},{BONOSTK,BONOSTK - 1}},1;
   8 ACTIVITY;
   9 FINDAR,1,NSN == SZ[1] && DC == XX[15],1,FORWARD,1,,{{STOCK,STOCK -
XX[1]},{BODAYS,BODAYS+(TNOW-BOSTART)}},IDX,1;
  10 ACTIVITY,,,,"HAVESTOCK";
  11 REVIEW5_ASSIGN_2: ASSIGN,{{STOCK,STOCK - TQTY}},1;
  12 ACTIVITY;
  13 REVIEW5_ASSIGN_1:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
}},1;
  14 ACTIVITY;
  15 REVIEW5_FINDAR_1: FINDAR,2,NSN == SZ[1] && QTY <= XX[1] && DCSEL ==
XX[15],1,FORWARD,1,"PROCBO",{{TQTY,QTY}},IDX,1;
  16 ACTIVITY,,0.0001,IDX > 0,"REVIEW5_ASSIGN_2";
  17 ACTIVITY,,,IDX == 0,"REVIEW5_TERMINATE_1";
  18 REVIEW5_TERMINATE_1: TERMINATE,INF;
  19 REVIEWRQ_ASSIGN_ERROR: ASSIGN,{{TREVIEWERROR,TSTOCK*MAX(-0.2, MIN(0.2,
RNORM(0,1,1)))},{TREVIEWLEVEL,MAX(CEIL(TINVPOS+TREVIEWERROR),0)}},1;
  20 ACTIVITY;
  21 REVIEW4_ASSIGN_8:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
},{STOCK,TSTOCK},{INVPOS,TINVPOS}},1;
  22 ACTIVITY,,,,"REVIEW4_FINDAR_2";
  23 REVIEW4_FINDAR_2: FINDAR,1,STRIB[1]==SZ[1] && DC == XX[15] && Q > 0 &&
TREVIEWLEVEL <= R,1,FORWARD,1,,{{ORDERS,ORDERS+1}},IDX,1;
  24 ACTIVITY,,,IDX > 0,"ORDER";
  25 ACTIVITY,,,IDX == 0,"REVIEW_TERMINATE_1";
  26 ORDER: GOON,1;
  27 ACTIVITY,,,,"REVIEW4_ASSIGN_4";
  28 REVIEW4_ASSIGN_4:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
}},1;
  29 ACTIVITY,,,,"REVIEW4_FINDAR_1";
  30 REVIEW4_FINDAR_1: FINDAR,1,STRIB[1]==SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{TLTIME,LEADTIME},{INVPOS,INVPOS +
Q},{ORDERQTYSUM,ORDERQTYSUM+Q}},IDX,2;
  31 ACTIVITY,,RLOGN(TLTIME,0.1,1),IDX>0,"REVIEW4_ASSIGN_2";
  32 ACTIVITY,,,IDX > 0,"REVIEW4_COLCT_1";
  33 ACTIVITY,,,IDX == 0,"ERROR_REVIEW";
  34 REVIEW4_ASSIGN_2:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
}},1;
  35 ACTIVITY,,,,"UPDATE_INV";
  36 UPDATE_INV: ASSIGN,{{TIVCD,TIVCD + UPRICE*Q*DCD},{TIV,TIV + UPRICE*Q}},1;
  37 ACTIVITY,,,DC == 1,"REVIEW4_ASSIGN_1";
  38 ACTIVITY,,,DC == 2,"NODE_9";
  39 ACTIVITY,,,DC == 3,"LDINV2_ASSIGN_8";
  40 REVIEW4_ASSIGN_1: ASSIGN,{{TIDC1,TIDC1 + (Q*UPRICE)}},1;
  41 ACTIVITY;
  42 REVIEW3_FINDAR_2: FINDAR,1,STRIB[1] == SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{STOCK,STOCK + Q},{TSTOCK,STOCK},{TINVPOS,INVPOS}},IDX,1;
  43 ACTIVITY,,,IDX > 0;
  44 ACTIVITY,,,IDX == 0,"REVIEW4_EVENT_1";
  45 ASSIGN,{{STOCK,TSTOCK},{INVPOS,TINVPOS}},1;
  46 ACTIVITY;
  47
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
}},1;
  48 ACTIVITY,,,,"REVIEW5_FINDAR_1";
  49 REVIEW4_EVENT_1: EVENT,5,1;
  50 ACTIVITY;
  51 TERMINATE,1;
  52 NODE_9: ASSIGN,{{TIDC2,TIDC2 + (Q*UPRICE)}},1;
  53 ACTIVITY,,,,"REVIEW3_FINDAR_2";
  54 LDINV2_ASSIGN_8: ASSIGN,{{TIDC3,TIDC3 + (Q*UPRICE)}},1;
  55 ACTIVITY,,,,"REVIEW3_FINDAR_2";
  56 REVIEW4_COLCT_1: COLCT,19,TLTIME,"Lead Time",,,,1;
  57 ACTIVITY;
```

```
  58 TERMINATE,INF;
  59 ERROR_REVIEW: TERMINATE,1;
  60 REVIEW_TERMINATE_1: TERMINATE,INF;
  61 COUNT_REVIEWS: FINDAR,1,STRIB[1]==SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{REVIEWS,REVIEWS+1},{STOCKLEVELSUM,STOCKLEVELSUM+STOCK},{TINVPOS,
INVPOS},{TSTOCK,STOCK}},IDX,1;
  62 ACTIVITY,,,ERRORRATE>0 && TSTOCK>20,"REVIEWRQ_ASSIGN_ERROR";
  63 ACTIVITY,,,ERRORRATE==0 || TSTOCK <= 20;
  64 REVIEWRQ_ASSIGN_NOERROR: ASSIGN,{{TREVIEWLEVEL,TINVPOS}},1;
  65 ACTIVITY,,,,"REVIEW4_ASSIGN_8";
  66 COUNT_REVIEWS_ASG: ASSIGN,{{SZ[1],STRIB[1]},{XX[15],DC}},1;
  67 ACTIVITY,,,,"COUNT_REVIEWS";
  68 REVIEWRQ: GOON,2;
  69 ACTIVITY,,REVIEWPERIOD,,"REVIEWRQ";
  70 ACTIVITY,,,,"COUNT_REVIEWS_ASG";

REVIEWRQ - Pass 3 successfully read

Reading network REVIEWSS - Pass 3...

   1 ;INVENTORY REVIEW (s,S)
   2 ;INV Entity incoming
   3 ;Individual Inventory Item is Reviewed
   4 ;DEMAND ENTITY
   5 SSPROCBO: GOON,1;
   6 ACTIVITY,,0.0001;
   7
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL},{XX[16],DCF},{BONOSTK,BONOSTK - 1}},1;
   8 ACTIVITY;
   9 FINDAR,1,NSN == SZ[1] && DC == XX[15],1,FORWARD,1,,{{STOCK,STOCK -
XX[1]},{BODAYS,BODAYS+(TNOW-BOSTART)}},IDX,2;
  10 ACTIVITY,,,,"HAVESTOCK";
  11 REVIEWSS_ASSIGN_ERROR: ASSIGN,{{TREVIEWERROR,TSTOCK*MAX(-0.2, MIN(0.2,
RNORM(0,1,1)))},{TREVIEWLEVEL,MAX(CEIL(INVPOS+TREVIEWERROR),0)}},1;
  12 ACTIVITY;
  13 REVIEWSS_ASSIGN_1:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],MAX(ATRIB[2],ATRIB[3
])},{XX[15],DC},{STOCK,TSTOCK},{INVPOS,TINVPOS}},1;
  14 ACTIVITY,,,,"REVIEWSS_FINDAR_1";
  15 REVIEWSS_FINDAR_1: FINDAR,1,STRIB[1]==SZ[1] && DC == XX[15] && Q > 0 &&
TREVIEWLEVEL <= R,1,FORWARD,1,,{{ORDERS,ORDERS+1}},IDX,1;
  16 ACTIVITY,,,IDX > 0,"ORDERSS";
  17 ACTIVITY,,,IDX == 0,"REVIEWSS_TERMINATE_1";
  18 ORDERSS: GOON,1;
  19 ACTIVITY,,,,"REVIEWSS_ASSIGN_2";
  20 REVIEWSS_ASSIGN_2:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
},{LASTORDERQ,MAX(Q-INVPOS,0)},{TQTY,LASTORDERQ}},1;
  21 ACTIVITY;
  22 REVIEWSS_FINDAR_2: FINDAR,1,STRIB[1]==SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{TLTIME,LEADTIME},{INVPOS,Q},{ORDERQTYSUM,ORDERQTYSUM+TQTY}},IDX,
2;
  23 ACTIVITY,,RLOGN(TLTIME,0.1,1),IDX>0,"REVIEWSS_ASSIGN_3";
  24 ACTIVITY,,,IDX > 0,"REVIEWSS_COLCT_1";
  25 ACTIVITY,,,IDX == 0,"SSERROR_REVIEW";
  26 REVIEWSS_ASSIGN_3:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
}},1;
  27 ACTIVITY,,,,"SSUPDATE_INV";
  28 SSUPDATE_INV: ASSIGN,{{TIVCD,TIVCD + UPRICE*LASTORDERQ*DCD},{TIV,TIV +
UPRICE*LASTORDERQ},{TQTY,LASTORDERQ}},1;
  29 ACTIVITY,,,DC == 1,"REVIEWSS_ASSIGN_4";
  30 ACTIVITY,,,DC == 2,"REVIEWSS_ASSIGN_5";
  31 ACTIVITY,,,DC == 3,"REVIEWSS_ASSIGN_6";
  32 REVIEWSS_ASSIGN_4: ASSIGN,{{TIDC1,TIDC1 + (LASTORDERQ*UPRICE)}},1;
  33 ACTIVITY;
  34 REVIEWSS_FINDAR_3: FINDAR,1,STRIB[1] == SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{STOCK,STOCK + TQTY},{TSTOCK,STOCK},{TINVPOS,INVPOS}},IDX,1;
  35 ACTIVITY,,,IDX > 0;
  36 ACTIVITY,,,IDX == 0,"REVIEWSS_EVENT_5";
```

```
  37 ASSIGN,{{STOCK,TSTOCK},{INVPOS,TINVPOS}},1;
  38 ACTIVITY;
  39
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
}},1;
  40 ACTIVITY;
  41 REVIEWSS_FINDAR_4: FINDAR,2,NSN == SZ[1] && QTY <= XX[1] && DCSEL ==
XX[15],1,FORWARD,1,"SSPROCBO",{{TQTY,QTY}},IDX,1;
  42 ACTIVITY,,0.0001,IDX > 0;
  43 ACTIVITY,,,IDX == 0,"REVIEWSS_TERMINATE_2";
  44 REVIEWSS_ASSIGN_7: ASSIGN,{{STOCK,STOCK - TQTY}},1;
  45 ACTIVITY;
  46 REVIEWSS_ASSIGN_8:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
}},1;
  47 ACTIVITY,,,,"REVIEWSS_FINDAR_4";
  48 REVIEWSS_TERMINATE_2: TERMINATE,INF;
  49 REVIEWSS_EVENT_5: EVENT,5,1;
  50 ACTIVITY;
  51 TERMINATE,1;
  52 REVIEWSS_ASSIGN_5: ASSIGN,{{TIDC2,TIDC2 + (LASTORDERQ*UPRICE)}},1;
  53 ACTIVITY,,,,"REVIEWSS_FINDAR_3";
  54 REVIEWSS_ASSIGN_6: ASSIGN,{{TIDC3,TIDC3 + (LASTORDERQ*UPRICE)}},1;
  55 ACTIVITY,,,,"REVIEWSS_FINDAR_3";
  56 REVIEWSS_COLCT_1: COLCT,19,TLTIME,"Lead Time",,,,1;
  57 ACTIVITY;
  58 TERMINATE,INF;
  59 SSERROR_REVIEW: TERMINATE,1;
  60 REVIEWSS_TERMINATE_1: TERMINATE,INF;
  61 SS_COUNT_REVIEWS: FINDAR,1,STRIB[1]==SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{REVIEWS,REVIEWS+1},{STOCKLEVELSUM,STOCKLEVELSUM+STOCK},{TINVPOS,
INVPOS},{TSTOCK,STOCK}},IDX,1;
  62 ACTIVITY,,,ERRORRATE>0 && TSTOCK > 20,"REVIEWSS_ASSIGN_ERROR";
  63 ACTIVITY,,,ERRORRATE==0 || TSTOCK <= 20;
  64 REVIEWSS_ASSIGN_NOERROR: ASSIGN,{{TREVIEWLEVEL,TINVPOS}},1;
  65 ACTIVITY,,,,"REVIEWSS_ASSIGN_1";
  66 SS_COUNT_REVIEWS_ASG: ASSIGN,{{SZ[1],STRIB[1]},{XX[15],DC}},1;
  67 ACTIVITY,,,,"SS_COUNT_REVIEWS";
  68 REVIEWSS: GOON,2;
  69 ACTIVITY,,REVIEWPERIOD,,"REVIEWSS";
  70 ACTIVITY,,,,"SS_COUNT_REVIEWS_ASG";

REVIEWSS - Pass 3 successfully read

Reading network STATS5 - Pass 3...

   1 ;STAT COLLECTION
   2 CREATE,30,30.0045,,INF,11;
   3 ACTIVITY;
   4 ACTIVITY,,,,"STATS_COLCT_1";
   5 ACTIVITY,,,,"STATS_COLCT_3";
   6 ACTIVITY,,,,"STATS_COLCT_4";
   7 ACTIVITY,,,,"STATS_COLCT_6";
   8 ACTIVITY,,,,"STATS_COLCT_7";
   9 ACTIVITY,,,,"STATS_COLCT_8";
  10 ACTIVITY,,,,"STATS2_COLCT_1";
  11 ACTIVITY,,,,"STATS_GOON_1";
  12 ACTIVITY,,,,"STATS5_ASSIGN_2";
  13 ACTIVITY,,,,"STATS_WRITE_1";
  14 COLCT,3,TRR,"TRR",,,,1;
  15 ACTIVITY;
  16 STATS_TERMINATE_1: TERMINATE,INF;
  17 STATS_COLCT_1: COLCT,2,TRS,"TRS",,,,1;
  18 ACTIVITY,,,,"STATS_TERMINATE_1";
  19 STATS_COLCT_3: COLCT,4,BO,"BO",,,,1;
  20 ACTIVITY,,,,"STATS_TERMINATE_1";
  21 STATS_COLCT_4: COLCT,5,BOPM,"BOPM",,,,1;
  22 ACTIVITY,,,,"STATS_TERMINATE_1";
  23 STATS_COLCT_6: COLCT,9,VMITOT,"TOTAL VMI",,,,1;
  24 ACTIVITY,,,,"STATS_TERMINATE_1";
  25 STATS_COLCT_7: COLCT,10,TRNPERMON,"TRANPERMON",,,,1;
```

```
  26 ACTIVITY,,,,"STATS_TERMINATE_1";
  27 STATS_COLCT_8: COLCT,11,NNQ(2),"BO_TRAN",,,,1;
  28 ACTIVITY,,,,"STATS_TERMINATE_1";
  29 STATS2_COLCT_1: COLCT,13,CDTRNPMON,"CDTRANPM",,,,1;
  30 ACTIVITY,,,,"STATS_TERMINATE_1";
  31 STATS_GOON_1: GOON,1;
  32 ACTIVITY,,,TRR + BOPM > 0;
  33 ACTIVITY,,,,"STATS_COLCT_5";
  34 STATS_COLCT_2: COLCT,1,TRS/(TRR+BOPM),"FR",,,,1;
  35 ACTIVITY;
  36 STATS_COLCT_5: COLCT,6,XX[6],"Mon Rec Val";
  37 ACTIVITY;
  38 ACTIVITY,,,,"STATS2_COLCT_3";
  39 ACTIVITY,,,,"STATS2_COLCT_4";
  40 ACTIVITY,,,,"STATS2_COLCT_5";
  41 COLCT,14,MFCDC1,"MonShipDC1",,,,1;
  42 ACTIVITY;
  43 STATS2_COLCT_2: COLCT,12,TRVCD,"CD_MonRecVal",,,,1;
  44 ACTIVITY,,,TRR>0;
  45 ACTIVITY,,0.02,,"STATS_ASSIGN_1";
  46 ASSIGN,{{POSOT,1-BO/TRR}},1;
  47 ACTIVITY;
  48 COLCT,8,POSOT,"Perc Ord Ship OnTime",,,,1;
  49 ACTIVITY,,.02;
  50 STATS_ASSIGN_1:
ASSIGN,{{TRR,0},{TRS,0},{BOPM,BO},{BO,0},{TRR1,0},{TRR2,0},{TRR3,0},{TRS1,0},{TRS2,0},
{TRS3,0},{DLNI,0},{DLDT,0},{DLB,0},{GBL,0},{NORDERS,0},{NREVIEWS,0}},1;
  51 ACTIVITY;
  52
ASSIGN,{{XX[6],0},{TRNPERMON,0},{TRVCD,0},{CDTRNPMON,0},{MFCDC3,0},{MFCDC2,0},{MFCDC1,
0},{NSEL1,0},{NSEL2,0},{NSEL3,0}},1;
  53 ACTIVITY;
  54 TERMINATE,INF;
  55 STATS2_COLCT_3: COLCT,15,MFCDC2,"MonShipDC2",,,,1;
  56 ACTIVITY,,,,"STATS2_COLCT_2";
  57 STATS2_COLCT_4: COLCT,16,MFCDC3,"MonShipDC3",,,,1;
  58 ACTIVITY,,,,"STATS2_COLCT_2";
  59 STATS2_COLCT_5: COLCT,17,MFCDC3 + MFCDC2 + MFCDC1,"MonShipTot",,,,1;
  60 ACTIVITY,,,,"STATS2_COLCT_2";
  61 STATS5_ASSIGN_2: ASSIGN,{{IDX2,1}},1;
  62 ACTIVITY;
  63 GETINVVAL:
FINDAR,1,LTRIB[4]==IDX2,1,FORWARD,1,,{{SZ[1],STRIB[1]},{NREVIEWS,REVIEWS*.025},{REVIEW
S,0},{NORDERS,ORDERS*0.5},{ORDERS,0},{NBACKORDERS,BACKORDERS*UPRICE*4},{BACKORDERS,0},
{NBODAYS,MIN(BODAYS/7.0,8)},{NSPORDERS,SPORDERS*40},{BODAYS,0},{SPORDERS,0},{TAVGSTOCK
LVL,NINT((STOCKLEVELSUM*REVIEWPERIOD)/30)},{TAVGINVVALUE,TAVGSTOCKLVL*UPRICE},{STOCKLE
VELSUM,0},{TORDERQTYVALUE,ORDERQTYSUM*UPRICE},{ORDERQTYSUM,0}},IDX,1;
  64 ACTIVITY,,,IDX>0&&DIRECTION>0;
  65 ACTIVITY,,,IDX>0&&DIRECTION<0,"STATS5_ASSIGN_1";
  66 ACTIVITY,,,IDX==0,"STATS5_TERMINATE_1";
  67 ASSIGN,{{COSTOUT,STRCAT(SZ[1],'.PCT')}},1;
  68 ACTIVITY;
  69 STATS5_ASSIGN_3: ASSIGN,{{IDX2,IDX2+1},{MCOST,NSPORDERS+(EXP(NBODAYS)-
1)*240+NBACKORDERS+(XX[28]*0.005)+NREVIEWS+NORDERS+(XX[28]*0.01)}},1;
  70 ACTIVITY;
  71 WRITECOSTS: EVENT,10,2;
  72 ACTIVITY,,,,"GETINVVAL";
  73 ACTIVITY;
  74 STATS5_TERMINATE_1: TERMINATE,INF;
  75 STATS5_ASSIGN_1: ASSIGN,{{COSTOUT,STRCAT(SZ[1],'.MCT')}},1;
  76 ACTIVITY,,,,"STATS5_ASSIGN_3";
  77 STATS_WRITE_1:
WRITE,CASEOUT,YES,"%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%
f,%f,%f,%f,%f,%f,%f,%f,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%f,%f,%f,%f,%f,%f,%d,%d,%d,%d\n",
{TNOW,CCAVG(1),CCAVG(2),CCAVG(3),CCAVG(4),CCAVG(5),CCAVG(6),CCAVG(7),CCAVG(8),CCAVG(9)
,CCAVG(10),CCAVG(11),CCAVG(12),CCAVG(13),TTAVG(1),TTAVG(2),TTAVG(3),TTAVG(4),0.0,0.0,0
.0,CCAVG(14),CCAVG(15),CCAVG(16),CCAVG(17),TTAVG(5),TTAVG(6),TTAVG(7),CCAVG(18),CCAVG(
19),BONOSTK,TRR1,TRR2,TRR3,TRS1,TRS2,TRS3,DLNI,DLDT,DLB,CCAVG(20),CCAVG(21),CCAVG(22),
CCAVG(23),CCAVG(24),CCAVG(25),GBL,NSEL1,NSEL2,NSEL3},1;
  78 ACTIVITY;
  79 TERMINATE,INF;
```

```
  80 ;Total Order value +(Special Order Cost)+(Idle Worker Cost)+(Initial Backorder
Cost)+(Holding Cost)+(Review Cost)+(Order Placement Cost)+(Cost of Capital)

STATS5 - Pass 3 successfully read


Translated network file PLUS.TRN successfully written
```

```
AweSim Input Translator, version 3.0
Copyright (C) 1999 Symix Systems, Inc.

Reading control CONMINUS ...

   1 ARRAY,1,3,{2,5,10};
   2 GEN,,,,1,YES,YES;
   3 LIMITS,38,37,5,16,11,1;
   4 PRIORITY,{{1,FIFO},{2,LVF(STRIB[1])}};
   5
EQUIVALENCE,{{NSN,STRIB[1]},{STOCK,ATRIB[1]},{Q,ATRIB[2]},{R,ATRIB[3]},{IDX,LL[4]},{UP
RICE,ATRIB[4]},{LEADTIME,ATRIB[5]},{LOC,LTRIB[1]},{CURDAY,XX[10]},{IDXREC,LL[6]},{INVP
OS,ATRIB[8]},{CD,LTRIB[2]},{DC,LTRIB[3]},{TIDC1,XX[17]},{TIDC2,XX[18]},{TIDC3,XX[19]},
{RECNUM,LTRIB[4]},{ORDERS,LTRIB[5]},{REVIEWS,LTRIB[6]},{BACKORDERS,ATRIB[14]},{BOSTART
,ATRIB[9]},{BODAYS,ATRIB[10]},{SPORDERS,LTRIB[7]},{REVIEWPERIOD,LTRIB[8]},{LASTORDERQ,
ATRIB[11]},{POLICY,LTRIB[10]},{STOCKLEVELSUM,ATRIB[12]},{ORDERQTYSUM,ATRIB[13]},{DDEST
,ATRIB[14]},{SDEST,ATRIB[15]},{DEMANDTYPE,LTRIB[11]},{ERRORRATE,ATRIB[16]}};
   6
EQUIVALENCE,{{QTY,ATRIB[1]},{DATE,ATRIB[2]},{SHIPTIME,ATRIB[3]},{DELTIME,ATRIB[4]},{IP
G,ATRIB[5]},{DUPRICE,ATRIB[6]},{STIMESAM,ATRIB[7]},{BOFLAG,LTRIB[1]},{TQ,ATRIB[8]},{DL
OC,LTRIB[2]},{DCD,LTRIB[3]},{DC1,LTRIB[4]},{F1,ATRIB[9]},{DC2,LTRIB[5]},{F2,ATRIB[10]}
,{DC3,LTRIB[6]},{F3,ATRIB[11]},{DCSEL,LTRIB[7]},{DCF,ATRIB[12]},{DLEADTIME,ATRIB[13]}}
;
   7
EQUIVALENCE,{{TRR,LL[1]},{TRS,LL[2]},{BO,LL[3]},{TIV,XX[5]},{BOPM,LL[5]},{POSOT,XX[8]}
,{COC,XX[9]},{TLTIME,XX[11]},{BONOSTK,LL[7]},{TEMPQ,XX[12]},{IDX2,LL[9]},{TEMPLOC,LL[1
0]},{VMITOT,LL[11]},{TRNPERMON,LL[12]},{TIVCD,XX[13]},{TRVCD,XX[14]},{TEMPCD,LL[13]},{
CDTRNPMON,LL[14]},{NUMDC,LL[15]},{MFCDC1,XX[20]},{MFCDC2,XX[21]},{MFCDC3,XX[22]},{TRSP
M,LL[16]},{STMON,LL[17]},{TRSHOLD,LL[18]},{IDX3,LL[19]},{TRR1,LL[20]},{TRR2,LL[21]},{T
RR3,LL[22]},{TRS1,LL[23]},{TRS2,LL[24]},{TRS3,LL[25]},{DLNI,LL[26]},{DLDT,LL[27]},{DLB
,LL[28]},{GBL,LL[29]},{NSEL1,LL[30]},{NSEL2,LL[31]},{NSEL3,LL[32]},{CASER,SZ[2]},{CASE
INV,SZ[3]},{CASEOUT,SZ[4]},{TSTOCK,XX[23]},{TINVPOS,XX[24]},{TQTY,XX[25]},{DIRECTION,L
L[35]},{COSTOUT,SZ[5]},{NREVIEWS,XX[30]},{NORDERS,XX[29]},{NBACKORDERS,XX[37]},{NBODAY
S,XX[31]},{NSPORDERS,LL[37]},{DELTA1,XX[32]},{DELTA2,XX[33]},{DELTA3,XX[34]},{TAVGSTOC
KLVL,LL[34]},{TAVGINVVALUE,XX[28]},{TORDERQTYVALUE,XX[36]},{TREVIEWERROR,XX[38]},{TREV
IEWLEVEL,LL[36]},{MCOST,XX[35]}};
   8 INTLC,{{COC,0.12},{NUMDC,1}};
   9 INITIALIZE,0.0,1096,YES,,NO;
  10 TIMST,1,TIV,"Total Inventory per Month",0,0.0,1.0;
  11 TIMST,4,TIVCD,"CostDriver Inventory per Month",0,0.0,1.0;
  12 TIMST,3,BO,"BO per month",0,0.0,1.0;
  13 TIMST,2,COC*TIV,"Cost of Cap",0,0.0,1.0;
  14 TIMST,5,TIDC1,"TotInvDC1",0,0.0,1.0;
  15 TIMST,6,TIDC2,"TotInvDC2",0,0.0,1.0;
  16 TIMST,7,TIDC3,"TotInvDC3",0,0.0,1.0;
  17 MONTR,SUMMARY,30.005,30;
  18 MONTR,CLEAR,30.01,30;
  19 NET;
  20
INTLC,{{CASER,"DEMAND.TXT"},{CASEINV,"INVENTORY1.TXT"},{CASEOUT,"COSTOUT_DEC.TXT"},{NU
MDC,1},{DIRECTION,-1}};
  21 FIN;

CONMINUS successfully read


Translated file MINUS successfully written
Reading network DEMAND5 - Pass 1...


DEMAND5 - Pass 1 successfully read

Reading network LDINVSP - Pass 1...


LDINVSP - Pass 1 successfully read

Reading network RECS5 - Pass 1...
```

```
RECS5 - Pass 1 successfully read

Reading network REVIEWRQ - Pass 1...

REVIEWRQ - Pass 1 successfully read

Reading network REVIEWSS - Pass 1...

REVIEWSS - Pass 1 successfully read

Reading network STATS5 - Pass 1...

STATS5 - Pass 1 successfully read

Reading network DEMAND5 - Pass 2...

DEMAND5 - Pass 2 successfully read

Reading network LDINVSP - Pass 2...

LDINVSP - Pass 2 successfully read

Reading network RECS5 - Pass 2...

RECS5 - Pass 2 successfully read

Reading network REVIEWRQ - Pass 2...

REVIEWRQ - Pass 2 successfully read

Reading network REVIEWSS - Pass 2...

REVIEWSS - Pass 2 successfully read

Reading network STATS5 - Pass 2...

STATS5 - Pass 2 successfully read

Reading network DEMAND5 - Pass 3...

   1 ;DEMAND FOR INVENTORY
   2 DEMAND: GOON,1;
   3 ACTIVITY;
   4 Samp_Shiptime: ASSIGN,{{DELTIME,DELTIME*0.9},{STIMESAM,SHIPTIME*0.9}},1;
   5 ACTIVITY,,,,"SELDC";
   6 DEMAND2:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL},{XX[16],DCF}},1;
   7 ACTIVITY;
   8 FINDAR,1,NSN == SZ[1] && STOCK >= XX[1] && DC ==
XX[15],1,FORWARD,1,,{{STOCK,STOCK - XX[1]},{INVPOS,INVPOS - XX[1]},{TEMPCD,CD}},IDX,1;
   9 ACTIVITY,,,IDX == 0,"BKORD";
  10 ACTIVITY,,,IDX > 0,"DEMAND5_ASSIGN_2";
  11 BKORD:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL},{XX[16],DCF}},1;
  12 ACTIVITY;
  13 CHECK: FINDAR,1,NSN == SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{TEMPLOC,LOC},{TLTIME,LEADTIME},{INVPOS,INVPOS -
XX[1]},{TEMPCD,CD},{BACKORDERS,BACKORDERS+XX[1]},{BOSTART,TNOW}},IDX,1;
  14 ACTIVITY;
```

```
  15 DEMAND_ASSIGN_3:
ASSIGN,{{DLOC,TEMPLOC},{DLEADTIME,TLTIME},{DCD,TEMPCD},{BOFLAG,1},{BONOSTK,BONOSTK +
1}},1;
  16 ACTIVITY,,,,"DEMAND5_ASSIGN_1";
  17 DEMAND5_ASSIGN_1:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL}},1;
  18 ACTIVITY;
  19 DEMAND5_FINDAR_1: FINDAR,1,NSN == SZ[1] && DC == XX[15] && Q ==
0,1,FORWARD,1,,{{SPORDERS,SPORDERS+1}},IDX,1;
  20 ACTIVITY,,,IDX > 0,"SPORD";
  21 ACTIVITY,,,IDX == 0;
  22 BORDER: EVENT,2,1;
  23 ACTIVITY;
  24 TERMINATE,INF;
  25 DEMAND5_ASSIGN_2: ASSIGN,{{DCD,TEMPCD}},2;
  26 ACTIVITY;
  27 HAVESTOCK:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL},{XX[16],DCF}},1;
  28 ACTIVITY;
  29 DEMAND4_FINDAR_1: FINDAR,1,NSN == SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{TEMPLOC,LOC},{TLTIME,LEADTIME}},IDX,2;
  30 ACTIVITY;
  31 DEMAND4_ASSIGN_5: ASSIGN,{{DLOC,TEMPLOC},{DLEADTIME,TLTIME}},1;
  32 ACTIVITY,,STIMESAM,,"DEMAND_ASSIGN_1";
  33 DEMAND_ASSIGN_1: ASSIGN,{{TIV,TIV - (QTY*DUPRICE)},{TRVCD,TRVCD +
QTY*DUPRICE*DCD},{XX[6],XX[6] + QTY*DUPRICE},{TIVCD,TIVCD - QTY*DUPRICE*DCD}},1;
  34 ACTIVITY,,,DCSEL== 1;
  35 ACTIVITY,,,DCSEL == 2,"LDINV2_ASSIGN_1";
  36 ACTIVITY,,,DCSEL == 3,"LDINV2_ASSIGN_2";
  37 ASSIGN,{{TIDC1,TIDC1 - (QTY*DUPRICE)},{MFCDC1,MFCDC1 + DCF}},1;
  38 ACTIVITY;
  39 DEMAND_COLCT_1: COLCT,7,TNOW-DATE,"Time To Ship",5,1,1,1;
  40 ACTIVITY,,,IPG==1;
  41 ACTIVITY,,,IPG==2,"DEMAND4_COLCT_5";
  42 ACTIVITY,,,IPG==3,"DEMAND4_COLCT_6";
  43 COLCT,20,TNOW - DATE,"TTS_IPG1",,,,1;
  44 ACTIVITY,,DELTIME;
  45 DEMAND4_COLCT_4: COLCT,18,TNOW - DATE,"Time To Deliver",5,1,1,1;
  46 ACTIVITY,,,IPG==1;
  47 ACTIVITY,,,IPG == 2,"DEMAND4_COLCT_2";
  48 ACTIVITY,,,IPG == 3,"DEMAND4_COLCT_3";
  49 COLCT,23,TNOW - DATE,"TTD-IPG1",,,,1;
  50 ACTIVITY;
  51 DEMAND4_GOON_1: GOON,1;
  52 ACTIVITY,,,TNOW - DATE <= ARRAY[1,IPG];
  53 ACTIVITY,,,TNOW - DATE > ARRAY[1,IPG],"DEMAND_ASSIGN_2";
  54 ACTIVITY,,,,"DEMAND4_EVENT_1";
  55 SHIPPED: ASSIGN,{{TRS,TRS + 1}},1;
  56 ACTIVITY,,,IPG == 1;
  57 ACTIVITY,,,IPG == 2,"DEMAND4_ASSIGN_1";
  58 ACTIVITY,,,IPG == 3,"DEMAND4_ASSIGN_2";
  59 ASSIGN,{{TRS1,TRS1 + 1}},1;
  60 ACTIVITY;
  61 DEMAND4_TERMINATE_1: TERMINATE,INF;
  62 DEMAND4_ASSIGN_1: ASSIGN,{{TRS2,TRS2 + 1}},1;
  63 ACTIVITY,,,,"DEMAND4_TERMINATE_1";
  64 DEMAND4_ASSIGN_2: ASSIGN,{{TRS3,TRS3 + 1}},1;
  65 ACTIVITY,,,,"DEMAND4_TERMINATE_1";
  66 DEMAND_ASSIGN_2: ASSIGN,{{BO,BO + 1}},1;
  67 ACTIVITY;
  68 GOON,1;
  69 ACTIVITY,,,BOFLAG == 1 && ((DELTIME + STIMESAM) > ARRAY[1,IPG]);
  70 ACTIVITY,,,BOFLAG == 1,"DEMAND4_ASSIGN_3";
  71 ACTIVITY,,,((DELTIME + STIMESAM) > ARRAY[1,IPG]),"DEMAND4_ASSIGN_4";
  72 ACTIVITY,,,,"DEMAND4_EVENT_2";
  73 ASSIGN,{{DLB,DLB + 1}},1;
  74 ACTIVITY,,,,"DEMAND4_TERMINATE_1";
  75 DEMAND4_ASSIGN_3: ASSIGN,{{DLNI,DLNI + 1}},1;
  76 ACTIVITY,,,,"DEMAND4_TERMINATE_1";
```

```
 77 DEMAND4_ASSIGN_4: ASSIGN,{{DLDT,DLDT + 1}},1;
 78 ACTIVITY,,,,"DEMAND4_TERMINATE_1";
 79 DEMAND4_EVENT_2: EVENT,7,1;
 80 ACTIVITY;
 81 TERMINATE,1;
 82 DEMAND4_EVENT_1: EVENT,6,1;
 83 ACTIVITY;
 84 TERMINATE,1;
 85 DEMAND4_COLCT_2: COLCT,24,TNOW - DATE,"TTD-IPG2",,,,1;
 86 ACTIVITY,,,,"DEMAND4_GOON_1";
 87 DEMAND4_COLCT_3: COLCT,25,TNOW - DATE,"TTD-IPG3",,,,1;
 88 ACTIVITY,,,,"DEMAND4_GOON_1";
 89 DEMAND4_COLCT_5: COLCT,21,TNOW - DATE,"TTS_IPG2",,,,1;
 90 ACTIVITY,,DELTIME,,"DEMAND4_COLCT_4";
 91 DEMAND4_COLCT_6: COLCT,22,TNOW - DATE,"TTS_IPG3",,,,1;
 92 ACTIVITY,,DELTIME,,"DEMAND4_COLCT_4";
 93 LDINV2_ASSIGN_1: ASSIGN,{{TIDC2,TIDC2 - (QTY*DUPRICE)},{MFCDC2,MFCDC2 + DCF}},1;
 94 ACTIVITY,,,,"DEMAND_COLCT_1";
 95 LDINV2_ASSIGN_2: ASSIGN,{{TIDC3,TIDC3 - (QTY*DUPRICE)},{MFCDC3,MFCDC3 + DCF}},1;
 96 ACTIVITY,,,,"DEMAND_COLCT_1";
 97 ;CHECK FOR STOCK ETC UPDATE
 98 ;BO Order-Demand entity only
 99 SPORD: GOON,1;
100 ACTIVITY;
101
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL},{XX[16],DCF}},1;
102 ACTIVITY;
103 FINDAR,1,STRIB[1] == SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{TLTIME,LEADTIME},{ORDERQTYSUM,ORDERQTYSUM+XX[1]}},IDX,2;
104 ACTIVITY,,RLOGN(TLTIME,0.1,1),IDX > 0;
105 ACTIVITY,,,IDX > 0,"DEMAND4_COLCT_1";
106
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL},{XX[16],DCF}},1;
107 ACTIVITY;
108 FINDAR,1,NSN == SZ[1] && DC == XX[15],1,FORWARD,1,,{{BODAYS,BODAYS+(TNOW-
BOSTART)},{INVPOS,INVPOS+XX[1]}},IDX,1;
109 ACTIVITY;
110 ASSIGN,{{BONOSTK,BONOSTK - 1},{TRVCD,TRVCD + QTY*DUPRICE*DCD},{XX[6],XX[6] +
QTY*DUPRICE}},1;
111 ACTIVITY,,,DCSEL== 1;
112 ACTIVITY,,,DCSEL == 2,"DEMAND_ASSIGN_10";
113 ACTIVITY,,,DCSEL == 3,"DEMAND_ASSIGN_9";
114 ASSIGN,{{MFCDC1,MFCDC1 + DCF}},1;
115 ACTIVITY;
116 DEMAND_COLCT_2: GOON,1;
117 ACTIVITY,,STIMESAM,,"DEMAND_COLCT_1";
118 DEMAND_ASSIGN_10: ASSIGN,{{MFCDC2,MFCDC2 + DCF}},1;
119 ACTIVITY,,,,"DEMAND_COLCT_2";
120 DEMAND_ASSIGN_9: ASSIGN,{{MFCDC3,MFCDC3 + DCF}},1;
121 ACTIVITY,,,,"DEMAND_COLCT_2";
122 DEMAND4_COLCT_1: COLCT,19,TLTIME,"Lead Time",,,,1;
123 ACTIVITY;
124 TERMINATE,INF;
125 ;Select DC to use
126 SELDC:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
1},{XX[16],F1}},1;
127 ACTIVITY;
128 FINDAR,1,NSN == SZ[1] && STOCK >= XX[1] && DC == XX[15],1,FORWARD,1,,,IDX,1;
129 ACTIVITY,,,IDX > 0;
130 ACTIVITY,,,IDX == 0 && NUMDC > 1,"DEMAND_ASSIGN_7";
131 ACTIVITY,,,IDX == 0 && NUMDC == 1,"DEMAND_ASSIGN_8";
132 ASSIGN,{{DCSEL,DC1},{DCF,F1},{NSEL1,NSEL1 + 1}},1;
133 ACTIVITY,,,,"DEMAND2";
134 DEMAND_ASSIGN_7:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
2},{XX[16],F2}},1;
135 ACTIVITY;
136 FINDAR,1,NSN == SZ[1] && STOCK >= XX[1] && DC == XX[15],1,FORWARD,1,,,IDX,1;
```

```
 137 ACTIVITY,,,IDX > 0;
 138 ACTIVITY,,,IDX == 0 && NUMDC > 2,"DEMAND_ASSIGN_5";
 139 ACTIVITY,,,IDX == 0 && NUMDC == 2,"DEMAND_ASSIGN_6";
 140 ASSIGN,{{DCSEL,DC2},{DCF,F2},{NSEL2,NSEL2 + 1}},1;
 141 ACTIVITY,,,,"DEMAND2";
 142 DEMAND_ASSIGN_5:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
3},{XX[16],F3}},1;
 143 ACTIVITY;
 144 FINDAR,1,NSN == SZ[1] && STOCK >= XX[1] && DC == XX[15],1,FORWARD,1,,,IDX,1;
 145 ACTIVITY,,,IDX > 0;
 146 ACTIVITY,,,IDX == 0,"DEMAND_ASSIGN_4";
 147 ASSIGN,{{DCSEL,DC3},{DCF,F3},{NSEL3,NSEL3 + 1}},1;
 148 ACTIVITY,,,,"DEMAND2";
 149 DEMAND_ASSIGN_4: ASSIGN,{{DCSEL,DC1},{DCF,F1},{NSEL1,NSEL1 + 1}},1;
 150 ACTIVITY,,,,"DEMAND2";
 151 DEMAND_ASSIGN_6: ASSIGN,{{DCSEL,DC1},{DCF,F1},{NSEL1,NSEL1 + 1}},1;
 152 ACTIVITY,,,,"DEMAND2";
 153 DEMAND_ASSIGN_8: ASSIGN,{{DCSEL,DC1},{DCF,F1},{NSEL1,NSEL1+1}},1;
 154 ACTIVITY,,,,"DEMAND2";

DEMAND5 - Pass 3 successfully read

Reading network LDINVSP - Pass 3...

   1 ;Load Inventory Files
   2 CREATE,INF,0.0,,1,1;
   3 ACTIVITY;
   4 NET1_READ_1:
READ,CASEINV,NO,IDX,,{RECNUM,NSN,STOCK,Q,R,REVIEWPERIOD,POLICY,UPRICE,LEADTIME,LOC,CD,
DC,DDEST,SDEST,DEMANDTYPE,ERRORRATE},1;
   5 ACTIVITY,,,IDX > 0;
   6 ACTIVITY,,,IDX == 0,"NET1_TERMINATE_1";
   7 ReadParams: EVENT,8,2;
   8 ACTIVITY;
   9 ACTIVITY,,,IDX2==1,"LDINVSP_ASSIGN_1";
  10 ACTIVITY,,,IDX2==2,"LDINVSP_ASSIGN_2";
  11 ACTIVITY,,,IDX2==3,"LDINVSP_ASSIGN_3";
  12
ASSIGN,{{STOCK,MAX(STOCK,0)},{INVPOS,STOCK},{ORDERS,0},{REVIEWS,0},{BACKORDERS,0},{STO
CKLEVELSUM,0},{ORDERQTYSUM,0}},1;
  13 ACTIVITY;
  14 LDINVSP_EVENT_1: EVENT,1,1;
  15 ACTIVITY,,,LOC > 1;
  16 ACTIVITY,,,LOC == 1,"LDINV_ASSIGN_6";
  17 LDINV_ASSIGN_1: ASSIGN,{{TIV,TIV + MAX(UPRICE*STOCK,0)}},1;
  18 ACTIVITY,,,DC == 1;
  19 ACTIVITY,,,DC == 2,"LDINV2_ASSIGN_9";
  20 ACTIVITY,,,DC == 3,"LDINV2_ASSIGN_7";
  21 ACTIVITY,,,DC < 1 || DC > 3,"LDINV2_EVENT_1";
  22 ASSIGN,{{TIDC1,TIDC1 + MAX(UPRICE*STOCK,0)}},1;
  23 ACTIVITY;
  24 LDINV2_GOON_1: GOON,2;
  25 ACTIVITY;
  26 ACTIVITY,,,,"LDINVSP_GOON_1";
  27 GOON,1;
  28 ACTIVITY,,,CD == 0,"NET1_READ_1";
  29 ACTIVITY,,,CD == 1;
  30 LDINV_ASSIGN_3: ASSIGN,{{TIVCD,TIVCD + MAX(UPRICE*STOCK,0)}},1;
  31 ACTIVITY,,,,"NET1_READ_1";
  32 LDINVSP_GOON_1: GOON,1;
  33 ACTIVITY,,UNFRM(0,(REVIEWPERIOD-1)),POLICY <= 4,"REVIEWRQ";
  34 ACTIVITY,,UNFRM(0,(REVIEWPERIOD-1)),POLICY >= 5,"REVIEWSS";
  35 ACTIVITY;
  36 PolicyError: EVENT,9,1;
  37 ACTIVITY;
  38 TERMINATE,INF;
  39 LDINV2_ASSIGN_9: ASSIGN,{{TIDC2,TIDC2 + MAX(UPRICE*STOCK,0)}},1;
  40 ACTIVITY,,,,"LDINV2_GOON_1";
  41 LDINV2_ASSIGN_7: ASSIGN,{{TIDC3,TIDC3 + MAX(UPRICE*STOCK,0)}},1;
  42 ACTIVITY,,,,"LDINV2_GOON_1";
```

```
  43 LDINV2_EVENT_1: EVENT,4,1;
  44 ACTIVITY;
  45 ERROR_wrong_DC: TERMINATE,1;
  46 LDINV_ASSIGN_6: ASSIGN,{{VMITOT,VMITOT + 1}},1;
  47 ACTIVITY,,,,"LDINV_ASSIGN_1";
  48 LDINVSP_ASSIGN_1: ASSIGN,{{Q,MAX(NINT(Q+(DIRECTION*XX[32]*XX[26])),0)}},1;
  49 ACTIVITY,,,,"LDINVSP_EVENT_1";
  50 LDINVSP_ASSIGN_2:
ASSIGN,{{Q,MAX(NINT(Q+(DIRECTION*XX[32]*XX[26])),0)},{R,MAX(NINT(R+(DIRECTION*XX[33]*X
X[26])),0)}},1;
  51 ACTIVITY,,,,"LDINVSP_EVENT_1";
  52 LDINVSP_ASSIGN_3:
ASSIGN,{{Q,MAX(NINT(Q+(DIRECTION*XX[32]*XX[26])),0)},{R,MAX(NINT(R+(DIRECTION*XX[33]*X
X[26])),0)},{REVIEWPERIOD,MAX(NINT(REVIEWPERIOD+(DIRECTION*XX[34]*XX[26])),1)}},1;
  53 ACTIVITY,,,,"LDINVSP_EVENT_1";
  54 NET1_TERMINATE_1: TERMINATE,INF;

LDINVSP - Pass 3 successfully read

Reading network RECS5 - Pass 3...

   1 ;RECS FOR INVENTORY
   2 CREATE,INF,0.001,,1,1;
   3 ACTIVITY;
   4 ASSIGN,{{IDXREC,1}},1;
   5 ACTIVITY;
   6 RECS_READ_1:
READ,CASER,NO,IDX,,{NSN,QTY,DATE,DELTIME,SHIPTIME,IPG,DUPRICE,DC1,F1,DC2,F2,DC3,F3},1;
   7 ACTIVITY,,,IDX > 0;
   8 ACTIVITY,,,IDX == 0,"RECS_TERMINATE_1";
   9 RECS_GOON_1: GOON,1;
  10 ACTIVITY,,DATE - TNOW,DATE == IDXREC;
  11 ACTIVITY,,1,DATE > IDXREC,"RECS_ASSIGN_2";
  12 RECS_ASSIGN_1: ASSIGN,{{TRR,TRR + 1}},2;
  13 ACTIVITY,,,,"RECS_READ_1";
  14 ACTIVITY;
  15 GOON,1;
  16 ACTIVITY,,,IPG == 1;
  17 ACTIVITY,,,IPG == 2,"RECS4_ASSIGN_3";
  18 ACTIVITY,,,IPG == 3,"RECS4_ASSIGN_4";
  19 ASSIGN,{{TRR1,TRR1 + 1}},1;
  20 ACTIVITY;
  21 RECS4_GOON_2: GOON,1;
  22 ACTIVITY;
  23 ASSIGN,{{SZ[1],STRIB[1]}},1;
  24 ACTIVITY;
  25 FINDAR,1,STRIB[1] == SZ[1],1,FORWARD,1,,,IDX3,1;
  26 ACTIVITY,,,IDX3 >0;
  27 ACTIVITY,,,IDX3 == 0,"ERROR";
  28 GOON,1;
  29 ACTIVITY,,,,"Demand";
  30 ERROR: EVENT,3,1;
  31 ACTIVITY;
  32 ERROR_No_Inv_Rec: TERMINATE,1;
  33 RECS4_ASSIGN_3: ASSIGN,{{TRR2,TRR2 + 1}},1;
  34 ACTIVITY,,,,"RECS4_GOON_2";
  35 RECS4_ASSIGN_4: ASSIGN,{{TRR3,TRR3 + 1}},1;
  36 ACTIVITY,,,,"RECS4_GOON_2";
  37 RECS_ASSIGN_2: ASSIGN,{{IDXREC,IDXREC + 1}},1;
  38 ACTIVITY,,,,"RECS_GOON_1";
  39 RECS_TERMINATE_1: TERMINATE,INF;

RECS5 - Pass 3 successfully read

Reading network REVIEWRQ - Pass 3...

   1 ;INVENTORY REVIEW (r, Q)
   2 ;Individual Inventory Item is Reviewed
   3 ;INV Entity incoming
   4 ;DEMAND ENTITY
   5 PROCBO: GOON,1;
```

```
   6 ACTIVITY,,0.0001;
   7
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL},{XX[16],DCF},{BONOSTK,BONOSTK - 1}},1;
   8 ACTIVITY;
   9 FINDAR,1,NSN == SZ[1] && DC == XX[15],1,FORWARD,1,,{{STOCK,STOCK -
XX[1]},{BODAYS,BODAYS+(TNOW-BOSTART)}},IDX,1;
  10 ACTIVITY,,,,"HAVESTOCK";
  11 REVIEW5_ASSIGN_2: ASSIGN,{{STOCK,STOCK - TQTY}},1;
  12 ACTIVITY;
  13 REVIEW5_ASSIGN_1:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
}},1;
  14 ACTIVITY;
  15 REVIEW5_FINDAR_1: FINDAR,2,NSN == SZ[1] && QTY <= XX[1] && DCSEL ==
XX[15],1,FORWARD,1,"PROCBO",{{TQTY,QTY}},IDX,1;
  16 ACTIVITY,,0.0001,IDX > 0,"REVIEW5_ASSIGN_2";
  17 ACTIVITY,,,IDX == 0,"REVIEW5_TERMINATE_1";
  18 REVIEW5_TERMINATE_1: TERMINATE,INF;
  19 REVIEWRQ_ASSIGN_ERROR: ASSIGN,{{TREVIEWERROR,TSTOCK*MAX(-0.2, MIN(0.2,
RNORM(0,1,1)))},{TREVIEWLEVEL,MAX(CEIL(TINVPOS+TREVIEWERROR),0)}},1;
  20 ACTIVITY;
  21 REVIEW4_ASSIGN_8:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
},{STOCK,TSTOCK},{INVPOS,TINVPOS}},1;
  22 ACTIVITY,,,,"REVIEW4_FINDAR_2";
  23 REVIEW4_FINDAR_2: FINDAR,1,STRIB[1]==SZ[1] && DC == XX[15] && Q > 0 &&
TREVIEWLEVEL <= R,1,FORWARD,1,,{{ORDERS,ORDERS+1}},IDX,1;
  24 ACTIVITY,,,IDX > 0,"ORDER";
  25 ACTIVITY,,,IDX == 0,"REVIEW_TERMINATE_1";
  26 ORDER: GOON,1;
  27 ACTIVITY,,,,"REVIEW4_ASSIGN_4";
  28 REVIEW4_ASSIGN_4:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
}},1;
  29 ACTIVITY,,,,"REVIEW4_FINDAR_1";
  30 REVIEW4_FINDAR_1: FINDAR,1,STRIB[1]==SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{TLTIME,LEADTIME},{INVPOS,INVPOS +
Q},{ORDERQTYSUM,ORDERQTYSUM+Q}},IDX,2;
  31 ACTIVITY,,RLOGN(TLTIME,0.1,1),IDX>0,"REVIEW4_ASSIGN_2";
  32 ACTIVITY,,,IDX > 0,"REVIEW4_COLCT_1";
  33 ACTIVITY,,,IDX == 0,"ERROR_REVIEW";
  34 REVIEW4_ASSIGN_2:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
}},1;
  35 ACTIVITY,,,,"UPDATE_INV";
  36 UPDATE_INV: ASSIGN,{{TIVCD,TIVCD + UPRICE*Q*DCD},{TIV,TIV + UPRICE*Q}},1;
  37 ACTIVITY,,,DC == 1,"REVIEW4_ASSIGN_1";
  38 ACTIVITY,,,DC == 2,"NODE_9";
  39 ACTIVITY,,,DC == 3,"LDINV2_ASSIGN_8";
  40 REVIEW4_ASSIGN_1: ASSIGN,{{TIDC1,TIDC1 + (Q*UPRICE)}},1;
  41 ACTIVITY;
  42 REVIEW3_FINDAR_2: FINDAR,1,STRIB[1] == SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{STOCK,STOCK + Q},{TSTOCK,STOCK},{TINVPOS,INVPOS}},IDX,1;
  43 ACTIVITY,,,IDX > 0;
  44 ACTIVITY,,,IDX == 0,"REVIEW4_EVENT_1";
  45 ASSIGN,{{STOCK,TSTOCK},{INVPOS,TINVPOS}},1;
  46 ACTIVITY;
  47
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
}},1;
  48 ACTIVITY,,,,"REVIEW5_FINDAR_1";
  49 REVIEW4_EVENT_1: EVENT,5,1;
  50 ACTIVITY;
  51 TERMINATE,1;
  52 NODE_9: ASSIGN,{{TIDC2,TIDC2 + (Q*UPRICE)}},1;
  53 ACTIVITY,,,,"REVIEW3_FINDAR_2";
  54 LDINV2_ASSIGN_8: ASSIGN,{{TIDC3,TIDC3 + (Q*UPRICE)}},1;
  55 ACTIVITY,,,,"REVIEW3_FINDAR_2";
  56 REVIEW4_COLCT_1: COLCT,19,TLTIME,"Lead Time",,,,1;
  57 ACTIVITY;
```

```
  58 TERMINATE,INF;
  59 ERROR_REVIEW: TERMINATE,1;
  60 REVIEW_TERMINATE_1: TERMINATE,INF;
  61 COUNT_REVIEWS: FINDAR,1,STRIB[1]==SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{REVIEWS,REVIEWS+1},{STOCKLEVELSUM,STOCKLEVELSUM+STOCK},{TINVPOS,
INVPOS},{TSTOCK,STOCK}},IDX,1;
  62 ACTIVITY,,,ERRORRATE>0 && TSTOCK>20,"REVIEWRQ_ASSIGN_ERROR";
  63 ACTIVITY,,,ERRORRATE==0 || TSTOCK <= 20;
  64 REVIEWRQ_ASSIGN_NOERROR: ASSIGN,{{TREVIEWLEVEL,TINVPOS}},1;
  65 ACTIVITY,,,,"REVIEW4_ASSIGN_8";
  66 COUNT_REVIEWS_ASG: ASSIGN,{{SZ[1],STRIB[1]},{XX[15],DC}},1;
  67 ACTIVITY,,,,"COUNT_REVIEWS";
  68 REVIEWRQ: GOON,2;
  69 ACTIVITY,,REVIEWPERIOD,,"REVIEWRQ";
  70 ACTIVITY,,,,"COUNT_REVIEWS_ASG";

REVIEWRQ - Pass 3 successfully read

Reading network REVIEWSS - Pass 3...

   1 ;INVENTORY REVIEW (s,S)
   2 ;INV Entity incoming
   3 ;Individual Inventory Item is Reviewed
   4 ;DEMAND ENTITY
   5 SSPROCBO: GOON,1;
   6 ACTIVITY,,0.0001;
   7
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL},{XX[16],DCF},{BONOSTK,BONOSTK - 1}},1;
   8 ACTIVITY;
   9 FINDAR,1,NSN == SZ[1] && DC == XX[15],1,FORWARD,1,,{{STOCK,STOCK -
XX[1]},{BODAYS,BODAYS+(TNOW-BOSTART)}},IDX,2;
  10 ACTIVITY,,,,"HAVESTOCK";
  11 REVIEWSS_ASSIGN_ERROR: ASSIGN,{{TREVIEWERROR,TSTOCK*MAX(-0.2, MIN(0.2,
RNORM(0,1,1)))},{TREVIEWLEVEL,MAX(CEIL(INVPOS+TREVIEWERROR),0)}},1;
  12 ACTIVITY;
  13 REVIEWSS_ASSIGN_1:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],MAX(ATRIB[2],ATRIB[3
])},{XX[15],DC},{STOCK,TSTOCK},{INVPOS,TINVPOS}},1;
  14 ACTIVITY,,,,"REVIEWSS_FINDAR_1";
  15 REVIEWSS_FINDAR_1: FINDAR,1,STRIB[1]==SZ[1] && DC == XX[15] && Q > 0 &&
TREVIEWLEVEL <= R,1,FORWARD,1,,{{ORDERS,ORDERS+1}},IDX,1;
  16 ACTIVITY,,,IDX > 0,"ORDERSS";
  17 ACTIVITY,,,IDX == 0,"REVIEWSS_TERMINATE_1";
  18 ORDERSS: GOON,1;
  19 ACTIVITY,,,,"REVIEWSS_ASSIGN_2";
  20 REVIEWSS_ASSIGN_2:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
},{LASTORDERQ,MAX(Q-INVPOS,0)},{TQTY,LASTORDERQ}},1;
  21 ACTIVITY;
  22 REVIEWSS_FINDAR_2: FINDAR,1,STRIB[1]==SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{TLTIME,LEADTIME},{INVPOS,Q},{ORDERQTYSUM,ORDERQTYSUM+TQTY}},IDX,
2;
  23 ACTIVITY,,RLOGN(TLTIME,0.1,1),IDX>0,"REVIEWSS_ASSIGN_3";
  24 ACTIVITY,,,IDX > 0,"REVIEWSS_COLCT_1";
  25 ACTIVITY,,,IDX == 0,"SSERROR_REVIEW";
  26 REVIEWSS_ASSIGN_3:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
}},1;
  27 ACTIVITY,,,,"SSUPDATE_INV";
  28 SSUPDATE_INV: ASSIGN,{{TIVCD,TIVCD + UPRICE*LASTORDERQ*DCD},{TIV,TIV +
UPRICE*LASTORDERQ},{TQTY,LASTORDERQ}},1;
  29 ACTIVITY,,,DC == 1,"REVIEWSS_ASSIGN_4";
  30 ACTIVITY,,,DC == 2,"REVIEWSS_ASSIGN_5";
  31 ACTIVITY,,,DC == 3,"REVIEWSS_ASSIGN_6";
  32 REVIEWSS_ASSIGN_4: ASSIGN,{{TIDC1,TIDC1 + (LASTORDERQ*UPRICE)}},1;
  33 ACTIVITY;
  34 REVIEWSS_FINDAR_3: FINDAR,1,STRIB[1] == SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{STOCK,STOCK + TQTY},{TSTOCK,STOCK},{TINVPOS,INVPOS}},IDX,1;
  35 ACTIVITY,,,IDX > 0;
  36 ACTIVITY,,,IDX == 0,"REVIEWSS_EVENT_5";
```

```
  37 ASSIGN,{{STOCK,TSTOCK},{INVPOS,TINVPOS}},1;
  38 ACTIVITY;
  39
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
}},1;
  40 ACTIVITY;
  41 REVIEWSS_FINDAR_4: FINDAR,2,NSN == SZ[1] && QTY <= XX[1] && DCSEL ==
XX[15],1,FORWARD,1,"SSPROCBO",{{TQTY,QTY}},IDX,1;
  42 ACTIVITY,,0.0001,IDX > 0;
  43 ACTIVITY,,,IDX == 0,"REVIEWSS_TERMINATE_2";
  44 REVIEWSS_ASSIGN_7: ASSIGN,{{STOCK,STOCK - TQTY}},1;
  45 ACTIVITY;
  46 REVIEWSS_ASSIGN_8:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
}},1;
  47 ACTIVITY,,,,"REVIEWSS_FINDAR_4";
  48 REVIEWSS_TERMINATE_2: TERMINATE,INF;
  49 REVIEWSS_EVENT_5: EVENT,5,1;
  50 ACTIVITY;
  51 TERMINATE,1;
  52 REVIEWSS_ASSIGN_5: ASSIGN,{{TIDC2,TIDC2 + (LASTORDERQ*UPRICE)}},1;
  53 ACTIVITY,,,,"REVIEWSS_FINDAR_3";
  54 REVIEWSS_ASSIGN_6: ASSIGN,{{TIDC3,TIDC3 + (LASTORDERQ*UPRICE)}},1;
  55 ACTIVITY,,,,"REVIEWSS_FINDAR_3";
  56 REVIEWSS_COLCT_1: COLCT,19,TLTIME,"Lead Time",,,,1;
  57 ACTIVITY;
  58 TERMINATE,INF;
  59 SSERROR_REVIEW: TERMINATE,1;
  60 REVIEWSS_TERMINATE_1: TERMINATE,INF;
  61 SS_COUNT_REVIEWS: FINDAR,1,STRIB[1]==SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{REVIEWS,REVIEWS+1},{STOCKLEVELSUM,STOCKLEVELSUM+STOCK},{TINVPOS,
INVPOS},{TSTOCK,STOCK}},IDX,1;
  62 ACTIVITY,,,ERRORRATE>0 && TSTOCK > 20,"REVIEWSS_ASSIGN_ERROR";
  63 ACTIVITY,,,ERRORRATE==0 || TSTOCK <= 20;
  64 REVIEWSS_ASSIGN_NOERROR: ASSIGN,{{TREVIEWLEVEL,TINVPOS}},1;
  65 ACTIVITY,,,,"REVIEWSS_ASSIGN_1";
  66 SS_COUNT_REVIEWS_ASG: ASSIGN,{{SZ[1],STRIB[1]},{XX[15],DC}},1;
  67 ACTIVITY,,,,"SS_COUNT_REVIEWS";
  68 REVIEWSS: GOON,2;
  69 ACTIVITY,,REVIEWPERIOD,,"REVIEWSS";
  70 ACTIVITY,,,,"SS_COUNT_REVIEWS_ASG";

REVIEWSS - Pass 3 successfully read

Reading network STATS5 - Pass 3...

   1 ;STAT COLLECTION
   2 CREATE,30,30.0045,,INF,11;
   3 ACTIVITY;
   4 ACTIVITY,,,,"STATS_COLCT_1";
   5 ACTIVITY,,,,"STATS_COLCT_3";
   6 ACTIVITY,,,,"STATS_COLCT_4";
   7 ACTIVITY,,,,"STATS_COLCT_6";
   8 ACTIVITY,,,,"STATS_COLCT_7";
   9 ACTIVITY,,,,"STATS_COLCT_8";
  10 ACTIVITY,,,,"STATS2_COLCT_1";
  11 ACTIVITY,,,,"STATS_GOON_1";
  12 ACTIVITY,,,,"STATS5_ASSIGN_2";
  13 ACTIVITY,,,,"STATS_WRITE_1";
  14 COLCT,3,TRR,"TRR",,,,1;
  15 ACTIVITY;
  16 STATS_TERMINATE_1: TERMINATE,INF;
  17 STATS_COLCT_1: COLCT,2,TRS,"TRS",,,,1;
  18 ACTIVITY,,,,"STATS_TERMINATE_1";
  19 STATS_COLCT_3: COLCT,4,BO,"BO",,,,1;
  20 ACTIVITY,,,,"STATS_TERMINATE_1";
  21 STATS_COLCT_4: COLCT,5,BOPM,"BOPM",,,,1;
  22 ACTIVITY,,,,"STATS_TERMINATE_1";
  23 STATS_COLCT_6: COLCT,9,VMITOT,"TOTAL VMI",,,,1;
  24 ACTIVITY,,,,"STATS_TERMINATE_1";
  25 STATS_COLCT_7: COLCT,10,TRNPERMON,"TRANPERMON",,,,1;
```

```
  26 ACTIVITY,,,,"STATS_TERMINATE_1";
  27 STATS_COLCT_8: COLCT,11,NNQ(2),"BO_TRAN",,,,1;
  28 ACTIVITY,,,,"STATS_TERMINATE_1";
  29 STATS2_COLCT_1: COLCT,13,CDTRNPMON,"CDTRANPM",,,,1;
  30 ACTIVITY,,,,"STATS_TERMINATE_1";
  31 STATS_GOON_1: GOON,1;
  32 ACTIVITY,,,TRR + BOPM > 0;
  33 ACTIVITY,,,,"STATS_COLCT_5";
  34 STATS_COLCT_2: COLCT,1,TRS/(TRR+BOPM),"FR",,,,1;
  35 ACTIVITY;
  36 STATS_COLCT_5: COLCT,6,XX[6],"Mon Rec Val";
  37 ACTIVITY;
  38 ACTIVITY,,,,"STATS2_COLCT_3";
  39 ACTIVITY,,,,"STATS2_COLCT_4";
  40 ACTIVITY,,,,"STATS2_COLCT_5";
  41 COLCT,14,MFCDC1,"MonShipDC1",,,,1;
  42 ACTIVITY;
  43 STATS2_COLCT_2: COLCT,12,TRVCD,"CD_MonRecVal",,,,1;
  44 ACTIVITY,,,TRR>0;
  45 ACTIVITY,,0.02,,"STATS_ASSIGN_1";
  46 ASSIGN,{{POSOT,1-BO/TRR}},1;
  47 ACTIVITY;
  48 COLCT,8,POSOT,"Perc Ord Ship OnTime",,,,1;
  49 ACTIVITY,,.02;
  50 STATS_ASSIGN_1:
ASSIGN,{{TRR,0},{TRS,0},{BOPM,BO},{BO,0},{TRR1,0},{TRR2,0},{TRR3,0},{TRS1,0},{TRS2,0},
{TRS3,0},{DLNI,0},{DLDT,0},{DLB,0},{GBL,0},{NORDERS,0},{NREVIEWS,0}},1;
  51 ACTIVITY;
  52
ASSIGN,{{XX[6],0},{TRNPERMON,0},{TRVCD,0},{CDTRNPMON,0},{MFCDC3,0},{MFCDC2,0},{MFCDC1,
0},{NSEL1,0},{NSEL2,0},{NSEL3,0}},1;
  53 ACTIVITY;
  54 TERMINATE,INF;
  55 STATS2_COLCT_3: COLCT,15,MFCDC2,"MonShipDC2",,,,1;
  56 ACTIVITY,,,,"STATS2_COLCT_2";
  57 STATS2_COLCT_4: COLCT,16,MFCDC3,"MonShipDC3",,,,1;
  58 ACTIVITY,,,,"STATS2_COLCT_2";
  59 STATS2_COLCT_5: COLCT,17,MFCDC3 + MFCDC2 + MFCDC1,"MonShipTot",,,,1;
  60 ACTIVITY,,,,"STATS2_COLCT_2";
  61 STATS5_ASSIGN_2: ASSIGN,{{IDX2,1}},1;
  62 ACTIVITY;
  63 GETINVVAL:
FINDAR,1,LTRIB[4]==IDX2,1,FORWARD,1,,{{SZ[1],STRIB[1]},{NREVIEWS,REVIEWS*.025},{REVIEW
S,0},{NORDERS,ORDERS*0.5},{ORDERS,0},{NBACKORDERS,BACKORDERS*UPRICE*4},{BACKORDERS,0},
{NBODAYS,MIN(BODAYS/7.0,8)},{NSPORDERS,SPORDERS*40},{BODAYS,0},{SPORDERS,0},{TAVGSTOCK
LVL,NINT((STOCKLEVELSUM*REVIEWPERIOD)/30)},{TAVGINVVALUE,TAVGSTOCKLVL*UPRICE},{STOCKLE
VELSUM,0},{TORDERQTYVALUE,ORDERQTYSUM*UPRICE},{ORDERQTYSUM,0}},IDX,1;
  64 ACTIVITY,,,IDX>0&&DIRECTION>0;
  65 ACTIVITY,,,IDX>0&&DIRECTION<0,"STATS5_ASSIGN_1";
  66 ACTIVITY,,,IDX==0,"STATS5_TERMINATE_1";
  67 ASSIGN,{{COSTOUT,STRCAT(SZ[1],'.PCT')}},1;
  68 ACTIVITY;
  69 STATS5_ASSIGN_3: ASSIGN,{{IDX2,IDX2+1},{MCOST,NSPORDERS+(EXP(NBODAYS)-
1)*240+NBACKORDERS+(XX[28]*0.005)+NREVIEWS+NORDERS+(XX[28]*0.01)}},1;
  70 ACTIVITY;
  71 WRITECOSTS: EVENT,10,2;
  72 ACTIVITY,,,,"GETINVVAL";
  73 ACTIVITY;
  74 STATS5_TERMINATE_1: TERMINATE,INF;
  75 STATS5_ASSIGN_1: ASSIGN,{{COSTOUT,STRCAT(SZ[1],'.MCT')}},1;
  76 ACTIVITY,,,,"STATS5_ASSIGN_3";
  77 STATS_WRITE_1:
WRITE,CASEOUT,YES,"%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%
f,%f,%f,%f,%f,%f,%f,%f,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%f,%f,%f,%f,%f,%f,%d,%d,%d,%d\n",
{TNOW,CCAVG(1),CCAVG(2),CCAVG(3),CCAVG(4),CCAVG(5),CCAVG(6),CCAVG(7),CCAVG(8),CCAVG(9)
,CCAVG(10),CCAVG(11),CCAVG(12),CCAVG(13),TTAVG(1),TTAVG(2),TTAVG(3),TTAVG(4),0.0,0.0,0
.0,CCAVG(14),CCAVG(15),CCAVG(16),CCAVG(17),TTAVG(5),TTAVG(6),TTAVG(7),CCAVG(18),CCAVG(
19),BONOSTK,TRR1,TRR2,TRR3,TRS1,TRS2,TRS3,DLNI,DLDT,DLB,CCAVG(20),CCAVG(21),CCAVG(22),
CCAVG(23),CCAVG(24),CCAVG(25),GBL,NSEL1,NSEL2,NSEL3},1;
  78 ACTIVITY;
  79 TERMINATE,INF;
```

```
  80 ;Total Order value +(Special Order Cost)+(Idle Worker Cost)+(Initial Backorder
Cost)+(Holding Cost)+(Review Cost)+(Order Placement Cost)+(Cost of Capital)

STATS5 - Pass 3 successfully read


Translated network file MINUS.TRN successfully written
```

```
AweSim Input Translator, version 3.0
Copyright (C) 1999 Symix Systems, Inc.

Reading control OPT ...

   1 ARRAY,1,3,{2,5,10};
   2 GEN,,,,10,YES,YES;
   3 LIMITS,38,37,5,16,11,1;
   4 PRIORITY,{{1,FIFO},{2,LVF(STRIB[1])}};
   5
EQUIVALENCE,{{NSN,STRIB[1]},{STOCK,ATRIB[1]},{Q,ATRIB[2]},{R,ATRIB[3]},{IDX,LL[4]},{UP
RICE,ATRIB[4]},{LEADTIME,ATRIB[5]},{LOC,LTRIB[1]},{CURDAY,XX[10]},{IDXREC,LL[6]},{INVP
OS,ATRIB[8]},{CD,LTRIB[2]},{DC,LTRIB[3]},{TIDC1,XX[17]},{TIDC2,XX[18]},{TIDC3,XX[19]},
{RECNUM,LTRIB[4]},{ORDERS,LTRIB[5]},{REVIEWS,LTRIB[6]},{BACKORDERS,ATRIB[14]},{BOSTART
,ATRIB[9]},{BODAYS,ATRIB[10]},{SPORDERS,LTRIB[7]},{REVIEWPERIOD,LTRIB[8]},{LASTORDERQ,
ATRIB[11]},{POLICY,LTRIB[10]},{STOCKLEVELSUM,ATRIB[12]},{ORDERQTYSUM,ATRIB[13]},{DDEST
,ATRIB[14]},{SDEST,ATRIB[15]},{DEMANDTYPE,LTRIB[11]},{ERRORRATE,ATRIB[16]}};
   6
EQUIVALENCE,{{QTY,ATRIB[1]},{DATE,ATRIB[2]},{SHIPTIME,ATRIB[3]},{DELTIME,ATRIB[4]},{IP
G,ATRIB[5]},{DUPRICE,ATRIB[6]},{STIMESAM,ATRIB[7]},{BOFLAG,LTRIB[1]},{TQ,ATRIB[8]},{DL
OC,LTRIB[2]},{DCD,LTRIB[3]},{DC1,LTRIB[4]},{F1,ATRIB[9]},{DC2,LTRIB[5]},{F2,ATRIB[10]}
,{DC3,LTRIB[6]},{F3,ATRIB[11]},{DCSEL,LTRIB[7]},{DCF,ATRIB[12]},{DLEADTIME,ATRIB[13]}}
;
   7 SEEDS,{{1783759,1,NO}};
   8
EQUIVALENCE,{{TRR,LL[1]},{TRS,LL[2]},{BO,LL[3]},{TIV,XX[5]},{BOPM,LL[5]},{POSOT,XX[8]}
,{COC,XX[9]},{TLTIME,XX[11]},{BONOSTK,LL[7]},{TEMPQ,XX[12]},{IDX2,LL[9]},{TEMPLOC,LL[1
0]},{VMITOT,LL[11]},{TRNPERMON,LL[12]},{TIVCD,XX[13]},{TRVCD,XX[14]},{TEMPCD,LL[13]},{
CDTRNPMON,LL[14]},{NUMDC,LL[15]},{MFCDC1,XX[20]},{MFCDC2,XX[21]},{MFCDC3,XX[22]},{TRSP
M,LL[16]},{STMON,LL[17]},{TRSHOLD,LL[18]},{IDX3,LL[19]},{TRR1,LL[20]},{TRR2,LL[21]},{T
RR3,LL[22]},{TRS1,LL[23]},{TRS2,LL[24]},{TRS3,LL[25]},{DLNI,LL[26]},{DLDT,LL[27]},{DLB
,LL[28]},{GBL,LL[29]},{NSEL1,LL[30]},{NSEL2,LL[31]},{NSEL3,LL[32]},{CASER,SZ[2]},{CASE
INV,SZ[3]},{CASEOUT,SZ[4]},{TSTOCK,XX[23]},{TINVPOS,XX[24]},{TQTY,XX[25]},{DIRECTION,L
L[35]},{COSTOUT,SZ[5]},{NREVIEWS,XX[30]},{NORDERS,XX[29]},{NBACKORDERS,XX[37]},{NBODAY
S,XX[31]},{NSPORDERS,LL[37]},{DELTA1,XX[32]},{DELTA2,XX[33]},{DELTA3,XX[34]},{MCOST,XX
[35]},{TAVGSTOCKLVL,LL[34]},{TAVGINVVALUE,XX[28]},{TORDERQTYVALUE,XX[36]},{TREVIEWERRO
R,XX[38]},{TREVIEWLEVEL,LL[36]}};
   9 INTLC,{{COC,0.12},{NUMDC,3}};
  10
INTLC,{{CASER,"DEMAND.TXT"},{CASEINV,"INVENTORYOPT.TXT"},{CASEOUT,"COSTOUT_OPT.TXT"},{
NUMDC,1}};
  11 INITIALIZE,0.0,1096,YES,,NO;
  12 TIMST,1,TIV,"Total Inventory per Month",0,0.0,1.0;
  13 TIMST,4,TIVCD,"CostDriver Inventory per Month",0,0.0,1.0;
  14 TIMST,3,BO,"BO per month",0,0.0,1.0;
  15 TIMST,2,COC*TIV,"Cost of Cap",0,0.0,1.0;
  16 TIMST,5,TIDC1,"TotInvDC1",0,0.0,1.0;
  17 TIMST,6,TIDC2,"TotInvDC2",0,0.0,1.0;
  18 TIMST,7,TIDC3,"TotInvDC3",0,0.0,1.0;
  19 MONTR,SUMMARY,30.005,30;
  20 MONTR,CLEAR,30.01,30;
  21 NET;
  22 FIN;

OPT successfully read


Translated file OPT successfully written
Reading network DEMAND5 - Pass 1...


DEMAND5 - Pass 1 successfully read

Reading network LDINV1ST - Pass 1...


LDINV1ST - Pass 1 successfully read

Reading network RECS1ST - Pass 1...
```

```
RECS1ST - Pass 1 successfully read

Reading network REVIEWRQ - Pass 1...

REVIEWRQ - Pass 1 successfully read

Reading network REVIEWSS - Pass 1...

REVIEWSS - Pass 1 successfully read

Reading network STATOPT - Pass 1...

STATOPT - Pass 1 successfully read

Reading network DEMAND5 - Pass 2...

DEMAND5 - Pass 2 successfully read

Reading network LDINV1ST - Pass 2...

LDINV1ST - Pass 2 successfully read

Reading network RECS1ST - Pass 2...

RECS1ST - Pass 2 successfully read

Reading network REVIEWRQ - Pass 2...

REVIEWRQ - Pass 2 successfully read

Reading network REVIEWSS - Pass 2...

REVIEWSS - Pass 2 successfully read

Reading network STATOPT - Pass 2...

STATOPT - Pass 2 successfully read

Reading network DEMAND5 - Pass 3...

   1 ;DEMAND FOR INVENTORY
   2 DEMAND: GOON,1;
   3 ACTIVITY;
   4 Samp_Shiptime: ASSIGN,{{DELTIME,DELTIME*0.9},{STIMESAM,SHIPTIME*0.9}},1;
   5 ACTIVITY,,,,"SELDC";
   6 DEMAND2:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL},{XX[16],DCF}},1;
   7 ACTIVITY;
   8 FINDAR,1,NSN == SZ[1] && STOCK >= XX[1] && DC ==
XX[15],1,FORWARD,1,,{{STOCK,STOCK - XX[1]},{INVPOS,INVPOS - XX[1]},{TEMPCD,CD}},IDX,1;
   9 ACTIVITY,,,IDX == 0,"BKORD";
  10 ACTIVITY,,,IDX > 0,"DEMAND5_ASSIGN_2";
  11 BKORD:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL},{XX[16],DCF}},1;
  12 ACTIVITY;
  13 CHECK: FINDAR,1,NSN == SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{TEMPLOC,LOC},{TLTIME,LEADTIME},{INVPOS,INVPOS -
XX[1]},{TEMPCD,CD},{BACKORDERS,BACKORDERS+XX[1]},{BOSTART,TNOW}},IDX,1;
  14 ACTIVITY;
```

```
  15 DEMAND_ASSIGN_3:
ASSIGN,{{DLOC,TEMPLOC},{DLEADTIME,TLTIME},{DCD,TEMPCD},{BOFLAG,1},{BONOSTK,BONOSTK +
1}},1;
  16 ACTIVITY,,,,"DEMAND5_ASSIGN_1";
  17 DEMAND5_ASSIGN_1:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL}},1;
  18 ACTIVITY;
  19 DEMAND5_FINDAR_1: FINDAR,1,NSN == SZ[1] && DC == XX[15] && Q ==
0,1,FORWARD,1,,{{SPORDERS,SPORDERS+1}},IDX,1;
  20 ACTIVITY,,,IDX > 0,"SPORD";
  21 ACTIVITY,,,IDX == 0;
  22 BORDER: EVENT,2,1;
  23 ACTIVITY;
  24 TERMINATE,INF;
  25 DEMAND5_ASSIGN_2: ASSIGN,{{DCD,TEMPCD}},2;
  26 ACTIVITY;
  27 HAVESTOCK:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL},{XX[16],DCF}},1;
  28 ACTIVITY;
  29 DEMAND4_FINDAR_1: FINDAR,1,NSN == SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{TEMPLOC,LOC},{TLTIME,LEADTIME}},IDX,2;
  30 ACTIVITY;
  31 DEMAND4_ASSIGN_5: ASSIGN,{{DLOC,TEMPLOC},{DLEADTIME,TLTIME}},1;
  32 ACTIVITY,,STIMESAM,,"DEMAND_ASSIGN_1";
  33 DEMAND_ASSIGN_1: ASSIGN,{{TIV,TIV - (QTY*DUPRICE)},{TRVCD,TRVCD +
QTY*DUPRICE*DCD},{XX[6],XX[6] + QTY*DUPRICE},{TIVCD,TIVCD - QTY*DUPRICE*DCD}},1;
  34 ACTIVITY,,,DCSEL== 1;
  35 ACTIVITY,,,DCSEL == 2,"LDINV2_ASSIGN_1";
  36 ACTIVITY,,,DCSEL == 3,"LDINV2_ASSIGN_2";
  37 ASSIGN,{{TIDC1,TIDC1 - (QTY*DUPRICE)},{MFCDC1,MFCDC1 + DCF}},1;
  38 ACTIVITY;
  39 DEMAND_COLCT_1: COLCT,7,TNOW-DATE,"Time To Ship",5,1,1,1;
  40 ACTIVITY,,,IPG==1;
  41 ACTIVITY,,,IPG==2,"DEMAND4_COLCT_5";
  42 ACTIVITY,,,IPG==3,"DEMAND4_COLCT_6";
  43 COLCT,20,TNOW - DATE,"TTS_IPG1",,,,1;
  44 ACTIVITY,,DELTIME;
  45 DEMAND4_COLCT_4: COLCT,18,TNOW - DATE,"Time To Deliver",5,1,1,1;
  46 ACTIVITY,,,IPG==1;
  47 ACTIVITY,,,IPG == 2,"DEMAND4_COLCT_2";
  48 ACTIVITY,,,IPG == 3,"DEMAND4_COLCT_3";
  49 COLCT,23,TNOW - DATE,"TTD-IPG1",,,,1;
  50 ACTIVITY;
  51 DEMAND4_GOON_1: GOON,1;
  52 ACTIVITY,,,TNOW - DATE <= ARRAY[1,IPG];
  53 ACTIVITY,,,TNOW - DATE > ARRAY[1,IPG],"DEMAND_ASSIGN_2";
  54 ACTIVITY,,,,"DEMAND4_EVENT_1";
  55 SHIPPED: ASSIGN,{{TRS,TRS + 1}},1;
  56 ACTIVITY,,,IPG == 1;
  57 ACTIVITY,,,IPG == 2,"DEMAND4_ASSIGN_1";
  58 ACTIVITY,,,IPG == 3,"DEMAND4_ASSIGN_2";
  59 ASSIGN,{{TRS1,TRS1 + 1}},1;
  60 ACTIVITY;
  61 DEMAND4_TERMINATE_1: TERMINATE,INF;
  62 DEMAND4_ASSIGN_1: ASSIGN,{{TRS2,TRS2 + 1}},1;
  63 ACTIVITY,,,,"DEMAND4_TERMINATE_1";
  64 DEMAND4_ASSIGN_2: ASSIGN,{{TRS3,TRS3 + 1}},1;
  65 ACTIVITY,,,,"DEMAND4_TERMINATE_1";
  66 DEMAND_ASSIGN_2: ASSIGN,{{BO,BO + 1}},1;
  67 ACTIVITY;
  68 GOON,1;
  69 ACTIVITY,,,BOFLAG == 1 && ((DELTIME + STIMESAM) > ARRAY[1,IPG]);
  70 ACTIVITY,,,BOFLAG == 1,"DEMAND4_ASSIGN_3";
  71 ACTIVITY,,,((DELTIME + STIMESAM) > ARRAY[1,IPG]),"DEMAND4_ASSIGN_4";
  72 ACTIVITY,,,,"DEMAND4_EVENT_2";
  73 ASSIGN,{{DLB,DLB + 1}},1;
  74 ACTIVITY,,,,"DEMAND4_TERMINATE_1";
  75 DEMAND4_ASSIGN_3: ASSIGN,{{DLNI,DLNI + 1}},1;
  76 ACTIVITY,,,,"DEMAND4_TERMINATE_1";
```

```
 77 DEMAND4_ASSIGN_4: ASSIGN,{{DLDT,DLDT + 1}},1;
 78 ACTIVITY,,,,"DEMAND4_TERMINATE_1";
 79 DEMAND4_EVENT_2: EVENT,7,1;
 80 ACTIVITY;
 81 TERMINATE,1;
 82 DEMAND4_EVENT_1: EVENT,6,1;
 83 ACTIVITY;
 84 TERMINATE,1;
 85 DEMAND4_COLCT_2: COLCT,24,TNOW - DATE,"TTD-IPG2",,,,1;
 86 ACTIVITY,,,,"DEMAND4_GOON_1";
 87 DEMAND4_COLCT_3: COLCT,25,TNOW - DATE,"TTD-IPG3",,,,1;
 88 ACTIVITY,,,,"DEMAND4_GOON_1";
 89 DEMAND4_COLCT_5: COLCT,21,TNOW - DATE,"TTS_IPG2",,,,1;
 90 ACTIVITY,,DELTIME,,"DEMAND4_COLCT_4";
 91 DEMAND4_COLCT_6: COLCT,22,TNOW - DATE,"TTS_IPG3",,,,1;
 92 ACTIVITY,,DELTIME,,"DEMAND4_COLCT_4";
 93 LDINV2_ASSIGN_1: ASSIGN,{{TIDC2,TIDC2 - (QTY*DUPRICE)},{MFCDC2,MFCDC2 + DCF}},1;
 94 ACTIVITY,,,,"DEMAND_COLCT_1";
 95 LDINV2_ASSIGN_2: ASSIGN,{{TIDC3,TIDC3 - (QTY*DUPRICE)},{MFCDC3,MFCDC3 + DCF}},1;
 96 ACTIVITY,,,,"DEMAND_COLCT_1";
 97 ;CHECK FOR STOCK ETC UPDATE
 98 ;BO Order-Demand entity only
 99 SPORD: GOON,1;
100 ACTIVITY;
101
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL},{XX[16],DCF}},1;
102 ACTIVITY;
103 FINDAR,1,STRIB[1] == SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{TLTIME,LEADTIME},{ORDERQTYSUM,ORDERQTYSUM+XX[1]}},IDX,2;
104 ACTIVITY,,RLOGN(TLTIME,0.1,1),IDX > 0;
105 ACTIVITY,,,IDX > 0,"DEMAND4_COLCT_1";
106
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL},{XX[16],DCF}},1;
107 ACTIVITY;
108 FINDAR,1,NSN == SZ[1] && DC == XX[15],1,FORWARD,1,,{{BODAYS,BODAYS+(TNOW-
BOSTART)},{INVPOS,INVPOS+XX[1]}},IDX,1;
109 ACTIVITY;
110 ASSIGN,{{BONOSTK,BONOSTK - 1},{TRVCD,TRVCD + QTY*DUPRICE*DCD},{XX[6],XX[6] +
QTY*DUPRICE}},1;
111 ACTIVITY,,,DCSEL== 1;
112 ACTIVITY,,,DCSEL == 2,"DEMAND_ASSIGN_10";
113 ACTIVITY,,,DCSEL == 3,"DEMAND_ASSIGN_9";
114 ASSIGN,{{MFCDC1,MFCDC1 + DCF}},1;
115 ACTIVITY;
116 DEMAND_COLCT_2: GOON,1;
117 ACTIVITY,,STIMESAM,,"DEMAND_COLCT_1";
118 DEMAND_ASSIGN_10: ASSIGN,{{MFCDC2,MFCDC2 + DCF}},1;
119 ACTIVITY,,,,"DEMAND_COLCT_2";
120 DEMAND_ASSIGN_9: ASSIGN,{{MFCDC3,MFCDC3 + DCF}},1;
121 ACTIVITY,,,,"DEMAND_COLCT_2";
122 DEMAND4_COLCT_1: COLCT,19,TLTIME,"Lead Time",,,,1;
123 ACTIVITY;
124 TERMINATE,INF;
125 ;Select DC to use
126 SELDC:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
1},{XX[16],F1}},1;
127 ACTIVITY;
128 FINDAR,1,NSN == SZ[1] && STOCK >= XX[1] && DC == XX[15],1,FORWARD,1,,,IDX,1;
129 ACTIVITY,,,IDX > 0;
130 ACTIVITY,,,IDX == 0 && NUMDC > 1,"DEMAND_ASSIGN_7";
131 ACTIVITY,,,IDX == 0 && NUMDC == 1,"DEMAND_ASSIGN_8";
132 ASSIGN,{{DCSEL,DC1},{DCF,F1},{NSEL1,NSEL1 + 1}},1;
133 ACTIVITY,,,,"DEMAND2";
134 DEMAND_ASSIGN_7:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
2},{XX[16],F2}},1;
135 ACTIVITY;
136 FINDAR,1,NSN == SZ[1] && STOCK >= XX[1] && DC == XX[15],1,FORWARD,1,,,IDX,1;
```

```
 137 ACTIVITY,,,IDX > 0;
 138 ACTIVITY,,,IDX == 0 && NUMDC > 2,"DEMAND_ASSIGN_5";
 139 ACTIVITY,,,IDX == 0 && NUMDC == 2,"DEMAND_ASSIGN_6";
 140 ASSIGN,{{DCSEL,DC2},{DCF,F2},{NSEL2,NSEL2 + 1}},1;
 141 ACTIVITY,,,,"DEMAND2";
 142 DEMAND_ASSIGN_5:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
3},{XX[16],F3}},1;
 143 ACTIVITY;
 144 FINDAR,1,NSN == SZ[1] && STOCK >= XX[1] && DC == XX[15],1,FORWARD,1,,,IDX,1;
 145 ACTIVITY,,,IDX > 0;
 146 ACTIVITY,,,IDX == 0,"DEMAND_ASSIGN_4";
 147 ASSIGN,{{DCSEL,DC3},{DCF,F3},{NSEL3,NSEL3 + 1}},1;
 148 ACTIVITY,,,,"DEMAND2";
 149 DEMAND_ASSIGN_4: ASSIGN,{{DCSEL,DC1},{DCF,F1},{NSEL1,NSEL1 + 1}},1;
 150 ACTIVITY,,,,"DEMAND2";
 151 DEMAND_ASSIGN_6: ASSIGN,{{DCSEL,DC1},{DCF,F1},{NSEL1,NSEL1 + 1}},1;
 152 ACTIVITY,,,,"DEMAND2";
 153 DEMAND_ASSIGN_8: ASSIGN,{{DCSEL,DC1},{DCF,F1},{NSEL1,NSEL1+1}},1;
 154 ACTIVITY,,,,"DEMAND2";

DEMAND5 - Pass 3 successfully read

Reading network LDINV1ST - Pass 3...

   1 ;Load Inventory Files
   2 CREATE,INF,0.0,,1,1;
   3 ACTIVITY;
   4 NET1_READ_1:
READ,CASEINV,YES,IDX,,{RECNUM,NSN,STOCK,Q,R,REVIEWPERIOD,POLICY,UPRICE,LEADTIME,LOC,CD
,DC,DDEST,SDEST,DEMANDTYPE,ERRORRATE},1;
   5 ACTIVITY,,,IDX > 0;
   6 ACTIVITY,,,IDX == 0,"NET1_TERMINATE_1";
   7
ASSIGN,{{STOCK,MAX(STOCK,0)},{INVPOS,STOCK},{ORDERS,0},{REVIEWS,0},{BACKORDERS,0},{STO
CKLEVELSUM,0},{SPORDERS,0},{BODAYS,0},{ORDERQTYSUM,0}},1;
   8 ACTIVITY;
   9 LDINVSP_EVENT_1: EVENT,1,1;
  10 ACTIVITY,,,LOC > 1;
  11 ACTIVITY,,,LOC == 1,"LDINV_ASSIGN_6";
  12 LDINV_ASSIGN_1: ASSIGN,{{TIV,TIV + MAX(UPRICE*STOCK,0)}},1;
  13 ACTIVITY,,,DC == 1;
  14 ACTIVITY,,,DC == 2,"LDINV2_ASSIGN_9";
  15 ACTIVITY,,,DC == 3,"LDINV2_ASSIGN_7";
  16 ACTIVITY,,,DC < 1 || DC > 3,"LDINV2_EVENT_1";
  17 ASSIGN,{{TIDC1,TIDC1 + MAX(UPRICE*STOCK,0)}},1;
  18 ACTIVITY;
  19 LDINV2_GOON_1: GOON,2;
  20 ACTIVITY;
  21 ACTIVITY,,,,"LDINVSP_GOON_1";
  22 GOON,1;
  23 ACTIVITY,,,CD == 0,"NET1_READ_1";
  24 ACTIVITY,,,CD == 1;
  25 LDINV_ASSIGN_3: ASSIGN,{{TIVCD,TIVCD + MAX(UPRICE*STOCK,0)}},1;
  26 ACTIVITY,,,,"NET1_READ_1";
  27 LDINVSP_GOON_1: GOON,1;
  28 ACTIVITY,,UNFRM(0,(REVIEWPERIOD-1)),POLICY <= 4,"REVIEWRQ";
  29 ACTIVITY,,UNFRM(0,(REVIEWPERIOD-1)),POLICY >= 5,"REVIEWSS";
  30 ACTIVITY;
  31 PolicyError: EVENT,9,1;
  32 ACTIVITY;
  33 TERMINATE,INF;
  34 LDINV2_ASSIGN_9: ASSIGN,{{TIDC2,TIDC2 + MAX(UPRICE*STOCK,0)}},1;
  35 ACTIVITY,,,,"LDINV2_GOON_1";
  36 LDINV2_ASSIGN_7: ASSIGN,{{TIDC3,TIDC3 + MAX(UPRICE*STOCK,0)}},1;
  37 ACTIVITY,,,,"LDINV2_GOON_1";
  38 LDINV2_EVENT_1: EVENT,4,1;
  39 ACTIVITY;
  40 ERROR_wrong_DC: TERMINATE,1;
  41 LDINV_ASSIGN_6: ASSIGN,{{VMITOT,VMITOT + 1}},1;
  42 ACTIVITY,,,,"LDINV_ASSIGN_1";
```

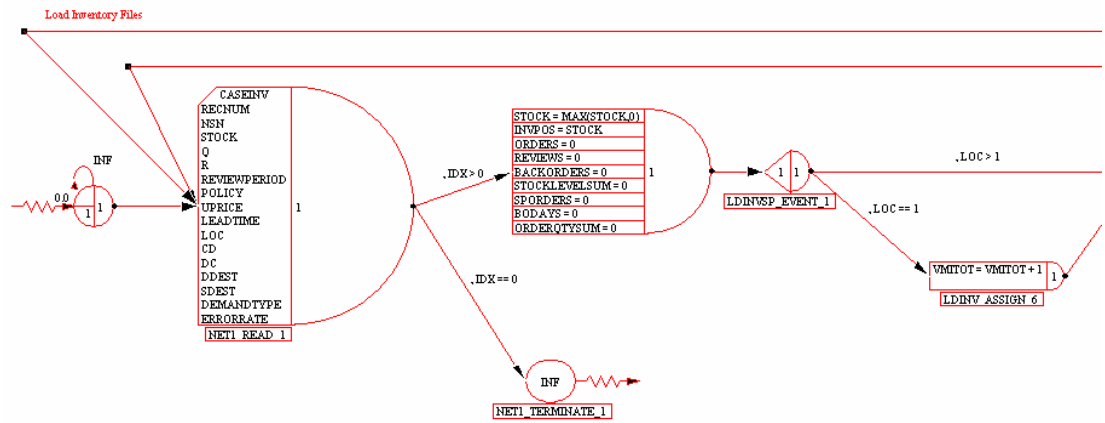
```
  43 NET1_TERMINATE_1: TERMINATE,INF;

LDINV1ST - Pass 3 successfully read

Reading network RECS1ST - Pass 3...

   1 ;RECS FOR INVENTORY
   2 CREATE,INF,0.001,,1,1;
   3 ACTIVITY;
   4 ASSIGN,{{IDXREC,1}},1;
   5 ACTIVITY;
   6 RECS_READ_1:
READ,CASER,YES,IDX,,{NSN,QTY,DATE,DELTIME,SHIPTIME,IPG,DUPRICE,DC1,F1,DC2,F2,DC3,F3},1
;
   7 ACTIVITY,,,IDX > 0;
   8 ACTIVITY,,,IDX == 0,"RECS_TERMINATE_1";
   9 RECS_GOON_1: GOON,1;
  10 ACTIVITY,,DATE - TNOW,DATE == IDXREC;
  11 ACTIVITY,,1,DATE > IDXREC,"RECS_ASSIGN_2";
  12 RECS_ASSIGN_1: ASSIGN,{{TRR,TRR + 1}},2;
  13 ACTIVITY,,,,"RECS_READ_1";
  14 ACTIVITY;
  15 GOON,1;
  16 ACTIVITY,,,IPG == 1;
  17 ACTIVITY,,,IPG == 2,"RECS4_ASSIGN_3";
  18 ACTIVITY,,,IPG == 3,"RECS4_ASSIGN_4";
  19 ASSIGN,{{TRR1,TRR1 + 1}},1;
  20 ACTIVITY;
  21 RECS4_GOON_2: GOON,1;
  22 ACTIVITY;
  23 ASSIGN,{{SZ[1],STRIB[1]}},1;
  24 ACTIVITY;
  25 FINDAR,1,STRIB[1] == SZ[1],1,FORWARD,1,,,IDX3,1;
  26 ACTIVITY,,,IDX3 >0;
  27 ACTIVITY,,,IDX3 == 0,"ERROR";
  28 GOON,1;
  29 ACTIVITY,,,,"Demand";
  30 ERROR: EVENT,3,1;
  31 ACTIVITY;
  32 ERROR_No_Inv_Rec: TERMINATE,1;
  33 RECS4_ASSIGN_3: ASSIGN,{{TRR2,TRR2 + 1}},1;
  34 ACTIVITY,,,,"RECS4_GOON_2";
  35 RECS4_ASSIGN_4: ASSIGN,{{TRR3,TRR3 + 1}},1;
  36 ACTIVITY,,,,"RECS4_GOON_2";
  37 RECS_ASSIGN_2: ASSIGN,{{IDXREC,IDXREC + 1}},1;
  38 ACTIVITY,,,,"RECS_GOON_1";
  39 RECS_TERMINATE_1: TERMINATE,INF;

RECS1ST - Pass 3 successfully read

Reading network REVIEWRQ - Pass 3...

   1 ;INVENTORY REVIEW (r, Q)
   2 ;Individual Inventory Item is Reviewed
   3 ;INV Entity incoming
   4 ;DEMAND ENTITY
   5 PROCBO: GOON,1;
   6 ACTIVITY,,0.0001;
   7
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL},{XX[16],DCF},{BONOSTK,BONOSTK - 1}},1;
   8 ACTIVITY;
   9 FINDAR,1,NSN == SZ[1] && DC == XX[15],1,FORWARD,1,,{{STOCK,STOCK -
XX[1]},{BODAYS,BODAYS+(TNOW-BOSTART)}},IDX,1;
  10 ACTIVITY,,,,"HAVESTOCK";
  11 REVIEW5_ASSIGN_2: ASSIGN,{{STOCK,STOCK - TQTY}},1;
  12 ACTIVITY;
  13 REVIEW5_ASSIGN_1:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
}},1;
  14 ACTIVITY;
```

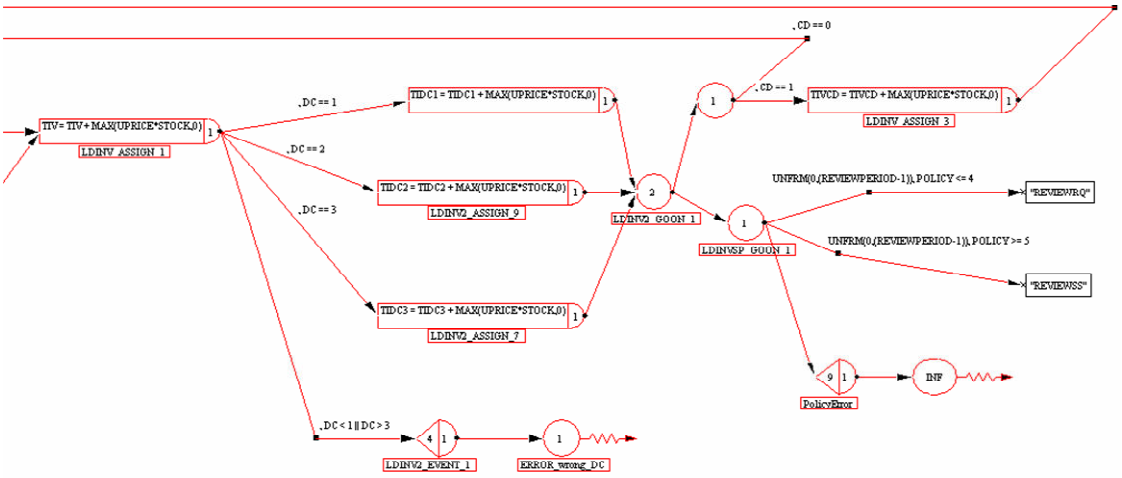
```
  15 REVIEW5_FINDAR_1: FINDAR,2,NSN == SZ[1] && QTY <= XX[1] && DCSEL ==
XX[15],1,FORWARD,1,"PROCBO",{{TQTY,QTY}},IDX,1;
  16 ACTIVITY,,0.0001,IDX > 0,"REVIEW5_ASSIGN_2";
  17 ACTIVITY,,,IDX == 0,"REVIEW5_TERMINATE_1";
  18 REVIEW5_TERMINATE_1: TERMINATE,INF;
  19 REVIEWRQ_ASSIGN_ERROR: ASSIGN,{{TREVIEWERROR,TSTOCK*MAX(-0.2, MIN(0.2,
RNORM(0,1,1)))},{TREVIEWLEVEL,MAX(CEIL(TINVPOS+TREVIEWERROR),0)}},1;
  20 ACTIVITY;
  21 REVIEW4_ASSIGN_8:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
},{STOCK,TSTOCK},{INVPOS,TINVPOS}},1;
  22 ACTIVITY,,,,"REVIEW4_FINDAR_2";
  23 REVIEW4_FINDAR_2: FINDAR,1,STRIB[1]==SZ[1] && DC == XX[15] && Q > 0 &&
TREVIEWLEVEL <= R,1,FORWARD,1,,{{ORDERS,ORDERS+1}},IDX,1;
  24 ACTIVITY,,,IDX > 0,"ORDER";
  25 ACTIVITY,,,IDX == 0,"REVIEW_TERMINATE_1";
  26 ORDER: GOON,1;
  27 ACTIVITY,,,,"REVIEW4_ASSIGN_4";
  28 REVIEW4_ASSIGN_4:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
}},1;
  29 ACTIVITY,,,,"REVIEW4_FINDAR_1";
  30 REVIEW4_FINDAR_1: FINDAR,1,STRIB[1]==SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{TLTIME,LEADTIME},{INVPOS,INVPOS +
Q},{ORDERQTYSUM,ORDERQTYSUM+Q}},IDX,2;
  31 ACTIVITY,,RLOGN(TLTIME,0.1,1),IDX>0,"REVIEW4_ASSIGN_2";
  32 ACTIVITY,,,IDX > 0,"REVIEW4_COLCT_1";
  33 ACTIVITY,,,IDX == 0,"ERROR_REVIEW";
  34 REVIEW4_ASSIGN_2:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
}},1;
  35 ACTIVITY,,,,"UPDATE_INV";
  36 UPDATE_INV: ASSIGN,{{TIVCD,TIVCD + UPRICE*Q*DCD},{TIV,TIV + UPRICE*Q}},1;
  37 ACTIVITY,,,DC == 1,"REVIEW4_ASSIGN_1";
  38 ACTIVITY,,,DC == 2,"NODE_9";
  39 ACTIVITY,,,DC == 3,"LDINV2_ASSIGN_8";
  40 REVIEW4_ASSIGN_1: ASSIGN,{{TIDC1,TIDC1 + (Q*UPRICE)}},1;
  41 ACTIVITY;
  42 REVIEW3_FINDAR_2: FINDAR,1,STRIB[1] == SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{STOCK,STOCK + Q},{TSTOCK,STOCK},{TINVPOS,INVPOS}},IDX,1;
  43 ACTIVITY,,,IDX > 0;
  44 ACTIVITY,,,IDX == 0,"REVIEW4_EVENT_1";
  45 ASSIGN,{{STOCK,TSTOCK},{INVPOS,TINVPOS}},1;
  46 ACTIVITY;
  47
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
}},1;
  48 ACTIVITY,,,,"REVIEW5_FINDAR_1";
  49 REVIEW4_EVENT_1: EVENT,5,1;
  50 ACTIVITY;
  51 TERMINATE,1;
  52 NODE_9: ASSIGN,{{TIDC2,TIDC2 + (Q*UPRICE)}},1;
  53 ACTIVITY,,,,"REVIEW3_FINDAR_2";
  54 LDINV2_ASSIGN_8: ASSIGN,{{TIDC3,TIDC3 + (Q*UPRICE)}},1;
  55 ACTIVITY,,,,"REVIEW3_FINDAR_2";
  56 REVIEW4_COLCT_1: COLCT,19,TLTIME,"Lead Time",,,,1;
  57 ACTIVITY;
  58 TERMINATE,INF;
  59 ERROR_REVIEW: TERMINATE,1;
  60 REVIEW_TERMINATE_1: TERMINATE,INF;
  61 COUNT_REVIEWS: FINDAR,1,STRIB[1]==SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{REVIEWS,REVIEWS+1},{STOCKLEVELSUM,STOCKLEVELSUM+STOCK},{TINVPOS,
INVPOS},{TSTOCK,STOCK}},IDX,1;
  62 ACTIVITY,,,ERRORRATE>0 && TSTOCK>20,"REVIEWRQ_ASSIGN_ERROR";
  63 ACTIVITY,,,ERRORRATE==0 || TSTOCK <= 20;
  64 REVIEWRQ_ASSIGN_NOERROR: ASSIGN,{{TREVIEWLEVEL,TINVPOS}},1;
  65 ACTIVITY,,,,"REVIEW4_ASSIGN_8";
  66 COUNT_REVIEWS_ASG: ASSIGN,{{SZ[1],STRIB[1]},{XX[15],DC}},1;
  67 ACTIVITY,,,,"COUNT_REVIEWS";
  68 REVIEWRQ: GOON,2;
  69 ACTIVITY,,REVIEWPERIOD,,"REVIEWRQ";
```

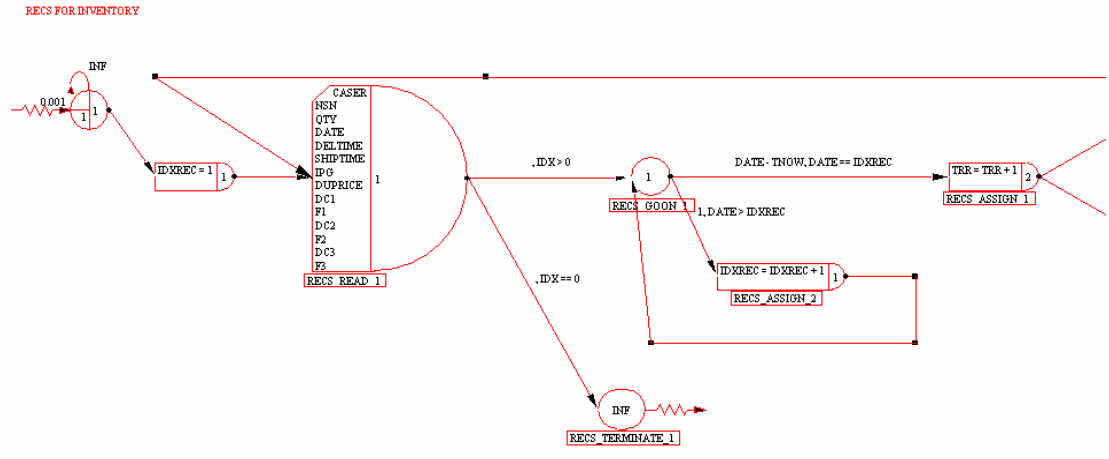
```
  70 ACTIVITY,,,,"COUNT_REVIEWS_ASG";

REVIEWRQ - Pass 3 successfully read

Reading network REVIEWSS - Pass 3...

   1 ;INVENTORY REVIEW (s,S)
   2 ;INV Entity incoming
   3 ;Individual Inventory Item is Reviewed
   4 ;DEMAND ENTITY
   5 SSPROCBO: GOON,1;
   6 ACTIVITY,,0.0001;
   7
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
SEL},{XX[16],DCF},{BONOSTK,BONOSTK - 1}},1;
   8 ACTIVITY;
   9 FINDAR,1,NSN == SZ[1] && DC == XX[15],1,FORWARD,1,,{{STOCK,STOCK -
XX[1]},{BODAYS,BODAYS+(TNOW-BOSTART)}},IDX,2;
  10 ACTIVITY,,,,"HAVESTOCK";
  11 REVIEWSS_ASSIGN_ERROR: ASSIGN,{{TREVIEWERROR,TSTOCK*MAX(-0.2, MIN(0.2,
RNORM(0,1,1)))},{TREVIEWLEVEL,MAX(CEIL(INVPOS+TREVIEWERROR),0)}},1;
  12 ACTIVITY;
  13 REVIEWSS_ASSIGN_1:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],MAX(ATRIB[2],ATRIB[3
])},{XX[15],DC},{STOCK,TSTOCK},{INVPOS,TINVPOS}},1;
  14 ACTIVITY,,,,"REVIEWSS_FINDAR_1";
  15 REVIEWSS_FINDAR_1: FINDAR,1,STRIB[1]==SZ[1] && DC == XX[15] && Q > 0 &&
TREVIEWLEVEL <= R,1,FORWARD,1,,{{ORDERS,ORDERS+1}},IDX,1;
  16 ACTIVITY,,,IDX > 0,"ORDERSS";
  17 ACTIVITY,,,IDX == 0,"REVIEWSS_TERMINATE_1";
  18 ORDERSS: GOON,1;
  19 ACTIVITY,,,,"REVIEWSS_ASSIGN_2";
  20 REVIEWSS_ASSIGN_2:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
},{LASTORDERQ,MAX(Q-INVPOS,0)},{TQTY,LASTORDERQ}},1;
  21 ACTIVITY;
  22 REVIEWSS_FINDAR_2: FINDAR,1,STRIB[1]==SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{TLTIME,LEADTIME},{INVPOS,Q},{ORDERQTYSUM,ORDERQTYSUM+TQTY}},IDX,
2;
  23 ACTIVITY,,RLOGN(TLTIME,0.1,1),IDX>0,"REVIEWSS_ASSIGN_3";
  24 ACTIVITY,,,IDX > 0,"REVIEWSS_COLCT_1";
  25 ACTIVITY,,,IDX == 0,"SSERROR_REVIEW";
  26 REVIEWSS_ASSIGN_3:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
}},1;
  27 ACTIVITY,,,,"SSUPDATE_INV";
  28 SSUPDATE_INV: ASSIGN,{{TIVCD,TIVCD + UPRICE*LASTORDERQ*DCD},{TIV,TIV +
UPRICE*LASTORDERQ},{TQTY,LASTORDERQ}},1;
  29 ACTIVITY,,,DC == 1,"REVIEWSS_ASSIGN_4";
  30 ACTIVITY,,,DC == 2,"REVIEWSS_ASSIGN_5";
  31 ACTIVITY,,,DC == 3,"REVIEWSS_ASSIGN_6";
  32 REVIEWSS_ASSIGN_4: ASSIGN,{{TIDC1,TIDC1 + (LASTORDERQ*UPRICE)}},1;
  33 ACTIVITY;
  34 REVIEWSS_FINDAR_3: FINDAR,1,STRIB[1] == SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{STOCK,STOCK + TQTY},{TSTOCK,STOCK},{TINVPOS,INVPOS}},IDX,1;
  35 ACTIVITY,,,IDX > 0;
  36 ACTIVITY,,,IDX == 0,"REVIEWSS_EVENT_5";
  37 ASSIGN,{{STOCK,TSTOCK},{INVPOS,TINVPOS}},1;
  38 ACTIVITY;
  39
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
}},1;
  40 ACTIVITY;
  41 REVIEWSS_FINDAR_4: FINDAR,2,NSN == SZ[1] && QTY <= XX[1] && DCSEL ==
XX[15],1,FORWARD,1,"SSPROCBO",{{TQTY,QTY}},IDX,1;
  42 ACTIVITY,,0.0001,IDX > 0;
  43 ACTIVITY,,,IDX == 0,"REVIEWSS_TERMINATE_2";
  44 REVIEWSS_ASSIGN_7: ASSIGN,{{STOCK,STOCK - TQTY}},1;
  45 ACTIVITY;
```

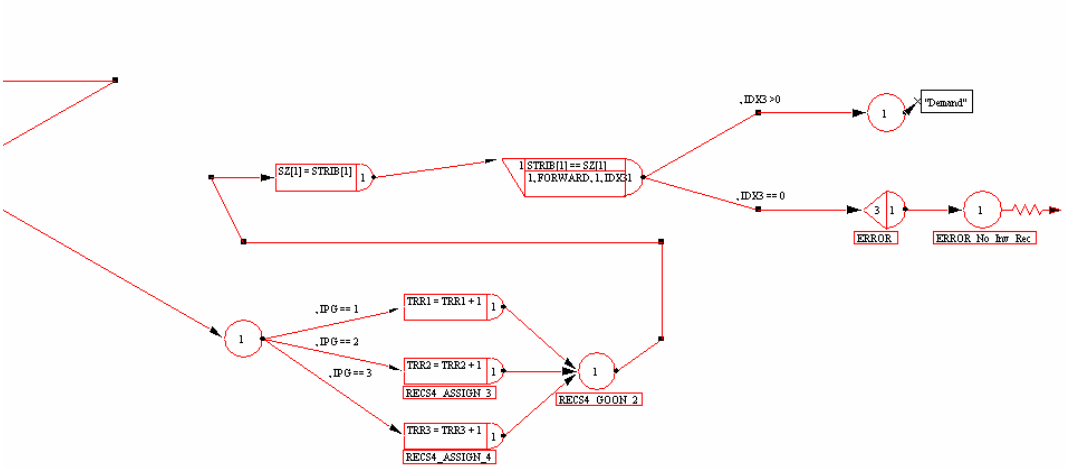
```
  46 REVIEWSS_ASSIGN_8:
ASSIGN,{{SZ[1],STRIB[1]},{XX[1],ATRIB[1]},{XX[2],ATRIB[2]},{XX[3],ATRIB[3]},{XX[15],DC
}},1;
  47 ACTIVITY,,,,"REVIEWSS_FINDAR_4";
  48 REVIEWSS_TERMINATE_2: TERMINATE,INF;
  49 REVIEWSS_EVENT_5: EVENT,5,1;
  50 ACTIVITY;
  51 TERMINATE,1;
  52 REVIEWSS_ASSIGN_5: ASSIGN,{{TIDC2,TIDC2 + (LASTORDERQ*UPRICE)}},1;
  53 ACTIVITY,,,,"REVIEWSS_FINDAR_3";
  54 REVIEWSS_ASSIGN_6: ASSIGN,{{TIDC3,TIDC3 + (LASTORDERQ*UPRICE)}},1;
  55 ACTIVITY,,,,"REVIEWSS_FINDAR_3";
  56 REVIEWSS_COLCT_1: COLCT,19,TLTIME,"Lead Time",,,,1;
  57 ACTIVITY;
  58 TERMINATE,INF;
  59 SSERROR_REVIEW: TERMINATE,1;
  60 REVIEWSS_TERMINATE_1: TERMINATE,INF;
  61 SS_COUNT_REVIEWS: FINDAR,1,STRIB[1]==SZ[1] && DC ==
XX[15],1,FORWARD,1,,{{REVIEWS,REVIEWS+1},{STOCKLEVELSUM,STOCKLEVELSUM+STOCK},{TINVPOS,
INVPOS},{TSTOCK,STOCK}},IDX,1;
  62 ACTIVITY,,,ERRORRATE>0 && TSTOCK > 20,"REVIEWSS_ASSIGN_ERROR";
  63 ACTIVITY,,,ERRORRATE==0 || TSTOCK <= 20;
  64 REVIEWSS_ASSIGN_NOERROR: ASSIGN,{{TREVIEWLEVEL,TINVPOS}},1;
  65 ACTIVITY,,,,"REVIEWSS_ASSIGN_1";
  66 SS_COUNT_REVIEWS_ASG: ASSIGN,{{SZ[1],STRIB[1]},{XX[15],DC}},1;
  67 ACTIVITY,,,,"SS_COUNT_REVIEWS";
  68 REVIEWSS: GOON,2;
  69 ACTIVITY,,REVIEWPERIOD,,"REVIEWSS";
  70 ACTIVITY,,,,"SS_COUNT_REVIEWS_ASG";

REVIEWSS - Pass 3 successfully read

Reading network STATOPT - Pass 3...

   1 ;STAT COLLECTION
   2 CREATE,30,30.0045,,INF,11;
   3 ACTIVITY;
   4 ACTIVITY,,,,"STATS_COLCT_1";
   5 ACTIVITY,,,,"STATS_COLCT_3";
   6 ACTIVITY,,,,"STATS_COLCT_4";
   7 ACTIVITY,,,,"STATS_COLCT_6";
   8 ACTIVITY,,,,"STATS_COLCT_7";
   9 ACTIVITY,,,,"STATS_COLCT_8";
  10 ACTIVITY,,,,"STATS2_COLCT_1";
  11 ACTIVITY,,,,"STATS_GOON_1";
  12 ACTIVITY,,,,"STATS5_ASSIGN_2";
  13 ACTIVITY,,,,"STATS_WRITE_1";
  14 COLCT,3,TRR,"TRR",,,,1;
  15 ACTIVITY;
  16 STATS_TERMINATE_1: TERMINATE,INF;
  17 STATS_COLCT_1: COLCT,2,TRS,"TRS",,,,1;
  18 ACTIVITY,,,,"STATS_TERMINATE_1";
  19 STATS_COLCT_3: COLCT,4,BO,"BO",,,,1;
  20 ACTIVITY,,,,"STATS_TERMINATE_1";
  21 STATS_COLCT_4: COLCT,5,BOPM,"BOPM",,,,1;
  22 ACTIVITY,,,,"STATS_TERMINATE_1";
  23 STATS_COLCT_6: COLCT,9,VMITOT,"TOTAL VMI",,,,1;
  24 ACTIVITY,,,,"STATS_TERMINATE_1";
  25 STATS_COLCT_7: COLCT,10,TRNPERMON,"TRANPERMON",,,,1;
  26 ACTIVITY,,,,"STATS_TERMINATE_1";
  27 STATS_COLCT_8: COLCT,11,NNQ(2),"BO_TRAN",,,,1;
  28 ACTIVITY,,,,"STATS_TERMINATE_1";
  29 STATS2_COLCT_1: COLCT,13,CDTRNPMON,"CDTRANPM",,,,1;
  30 ACTIVITY,,,,"STATS_TERMINATE_1";
  31 STATS_GOON_1: GOON,1;
  32 ACTIVITY,,,TRR + BOPM > 0;
  33 ACTIVITY,,,,"STATS_COLCT_5";
  34 STATS_COLCT_2: COLCT,1,TRS/(TRR+BOPM),"FR",,,,1;
  35 ACTIVITY;
  36 STATS_COLCT_5: COLCT,6,XX[6],"Mon Rec Val";
  37 ACTIVITY;
```

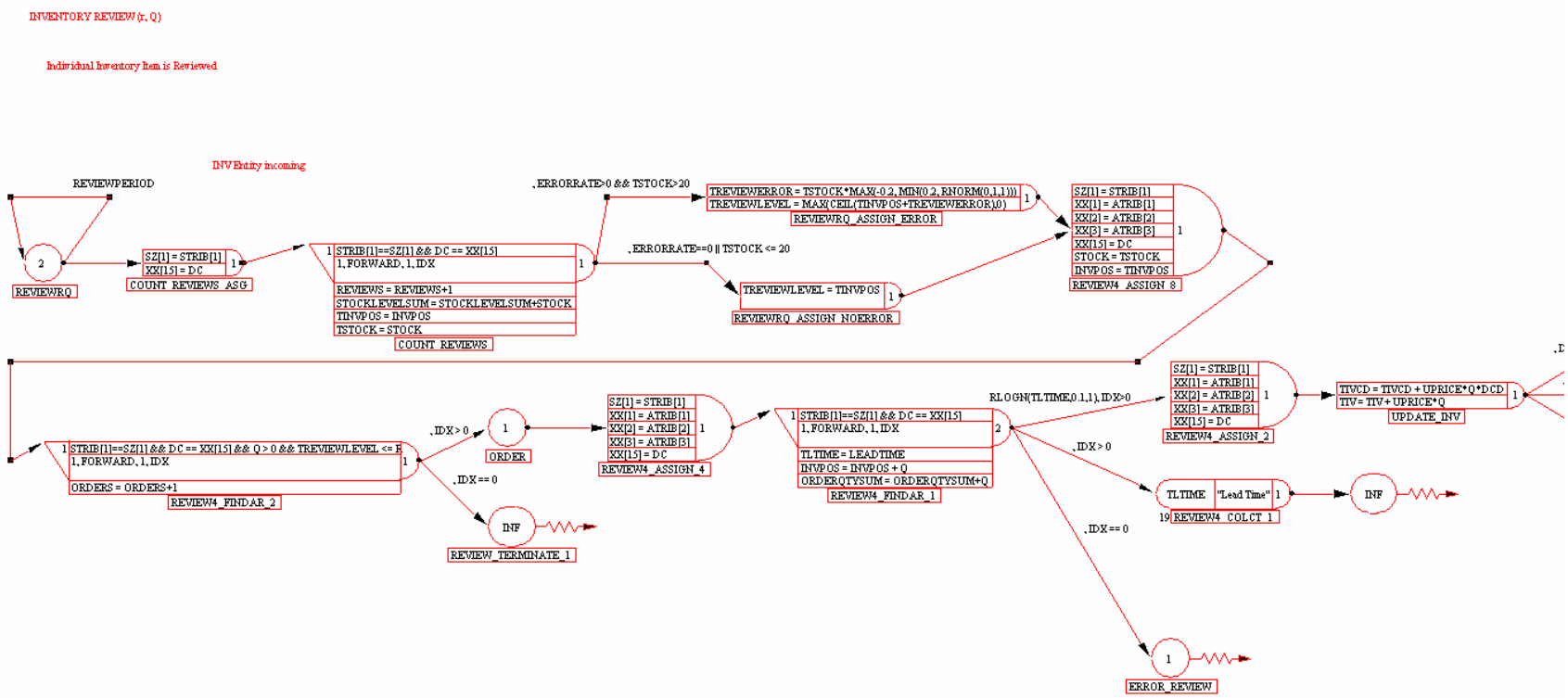
```
  38 ACTIVITY,,,,"STATS2_COLCT_3";
  39 ACTIVITY,,,,"STATS2_COLCT_4";
  40 ACTIVITY,,,,"STATS2_COLCT_5";
  41 COLCT,14,MFCDC1,"MonShipDC1",,,,1;
  42 ACTIVITY;
  43 STATS2_COLCT_2: COLCT,12,TRVCD,"CD_MonRecVal",,,,1;
  44 ACTIVITY,,,TRR>0;
  45 ACTIVITY,,0.02,,"STATS_ASSIGN_1";
  46 ASSIGN,{{POSOT,1-BO/TRR}},1;
  47 ACTIVITY;
  48 COLCT,8,POSOT,"Perc Ord Ship OnTime",,,,1;
  49 ACTIVITY,,.02;
  50 STATS_ASSIGN_1:
ASSIGN,{{TRR,0},{TRS,0},{BOPM,BO},{BO,0},{TRR1,0},{TRR2,0},{TRR3,0},{TRS1,0},{TRS2,0},
{TRS3,0},{DLNI,0},{DLDT,0},{DLB,0},{GBL,0}},1;
  51 ACTIVITY;
  52
ASSIGN,{{XX[6],0},{TRNPERMON,0},{TRVCD,0},{CDTRNPMON,0},{MFCDC3,0},{MFCDC2,0},{MFCDC1,
0},{NSEL1,0},{NSEL2,0},{NSEL3,0}},1;
  53 ACTIVITY;
  54 TERMINATE,INF;
  55 STATS2_COLCT_3: COLCT,15,MFCDC2,"MonShipDC2",,,,1;
  56 ACTIVITY,,,,"STATS2_COLCT_2";
  57 STATS2_COLCT_4: COLCT,16,MFCDC3,"MonShipDC3",,,,1;
  58 ACTIVITY,,,,"STATS2_COLCT_2";
  59 STATS2_COLCT_5: COLCT,17,MFCDC3 + MFCDC2 + MFCDC1,"MonShipTot",,,,1;
  60 ACTIVITY,,,,"STATS2_COLCT_2";
  61 STATS5_ASSIGN_2: ASSIGN,{{IDX2,1}},1;
  62 ACTIVITY;
  63 GETINVVAL:
FINDAR,1,LTRIB[4]==IDX2,1,FORWARD,1,,{{SZ[1],STRIB[1]},{NREVIEWS,REVIEWS*0.25},{REVIEW
S,0},{NORDERS,ORDERS*0.5},{ORDERS,0},{NBACKORDERS,BACKORDERS*UPRICE*4},{BACKORDERS,0},
{NBODAYS,MIN(BODAYS/7.0,8)},{NSPORDERS,SPORDERS*40},{BODAYS,0},{SPORDERS,0},{TAVGSTOCK
LVL,NINT((STOCKLEVELSUM*REVIEWPERIOD)/30)},{TAVGINVVALUE,TAVGSTOCKLVL*UPRICE},{STOCKLE
VELSUM,0},{TORDERQTYVALUE,ORDERQTYSUM*UPRICE},{ORDERQTYSUM,0}},IDX,1;
  64 ACTIVITY,,,IDX>0;
  65 ACTIVITY,,,IDX==0,"STATS5_TERMINATE_1";
  66
ASSIGN,{{COSTOUT,STRCAT(SZ[1],'OPT.CST')},{IDX2,IDX2+1},{MCOST,NSPORDERS+(EXP(NBODAYS)
-1)*240+NBACKORDERS+(XX[28]*0.005)+NREVIEWS+NORDERS+(XX[28]*0.01)}},1;
  67 ACTIVITY;
  68 WRITECOSTS: EVENT,10,2;
  69 ACTIVITY,,,,"GETINVVAL";
  70 ACTIVITY;
  71 STATS5_TERMINATE_1: TERMINATE,INF;
  72 STATS_WRITE_1:
WRITE,CASEOUT,YES,"%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%
f,%f,%f,%f,%f,%f,%f,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%f,%f,%f,%f,%f,%f,%d,%d,%d,%d\n",
{TNOW,CCAVG(1),CCAVG(2),CCAVG(3),CCAVG(4),CCAVG(5),CCAVG(6),CCAVG(7),CCAVG(8),CCAVG(9)
,CCAVG(10),CCAVG(11),CCAVG(12),CCAVG(13),TTAVG(1),TTAVG(2),TTAVG(3),TTAVG(4),0.0,0.0,0
.0,CCAVG(14),CCAVG(15),CCAVG(16),CCAVG(17),TTAVG(5),TTAVG(6),TTAVG(7),CCAVG(18),CCAVG(
19),BONOSTK,TRR1,TRR2,TRR3,TRS1,TRS2,TRS3,DLNI,DLDT,DLB,CCAVG(20),CCAVG(21),CCAVG(22),
CCAVG(23),CCAVG(24),CCAVG(25),GBL,NSEL1,NSEL2,NSEL3},1;
  73 ACTIVITY;
  74 TERMINATE,INF;
  75 ;Order value + (Special Order Cost)+(Idle Worker Cost)+(Initial Backorder
Cost)+(Holding Cost)+(Review Cost)+(Order Placement Cost)+(Cost of Capital)

STATOPT - Pass 3 successfully read


Translated network file OPT.TRN successfully written
```

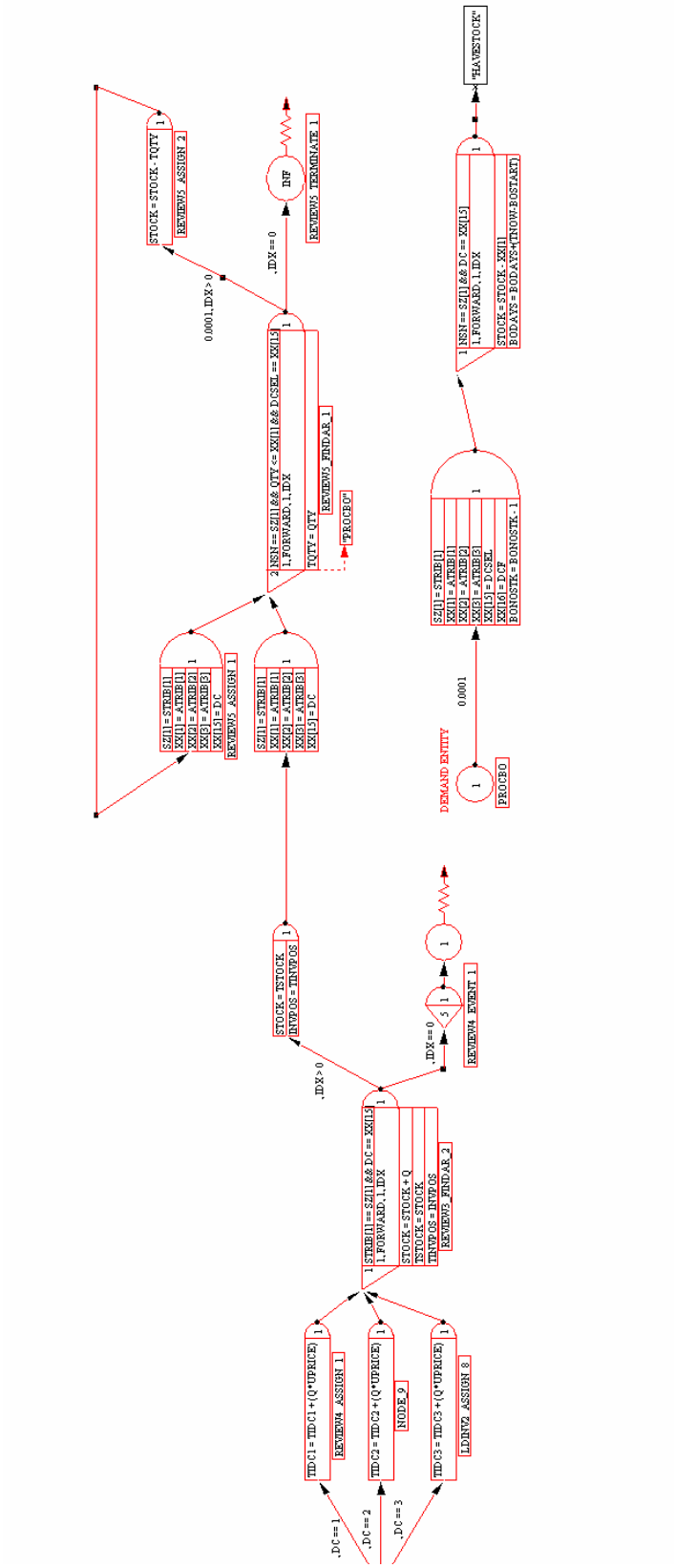

Load Inventory Files
INF
0.0
1
CASEINV
RECNUM
NSN
STOCK
Q
R
REVIEWPERIOD
POLICY
UPRICE
LEADTIME
LOC
CD
DC
DDEST
SDEST
DEMANDTYPE
ERRORRATE
NET1_READ_1
,IDX > 0
,IDX == 0
STOCK = MAX(STOCK,0)
INVPOS = STOCK
ORDERS = 0
REVIEWS = 0
BACKORDERS = 0
STOCKLEVELSUM = 0
SPORDERS = 0
BODAYS = 0
ORDERQTYSUM = 0
LDINVSP_EVENT_1
,LOC > 1
,LOC == 1
VMITOT = VMITOT + 1
LDINV_ASSIGN_6
INF
NET1_TERMINATE_1

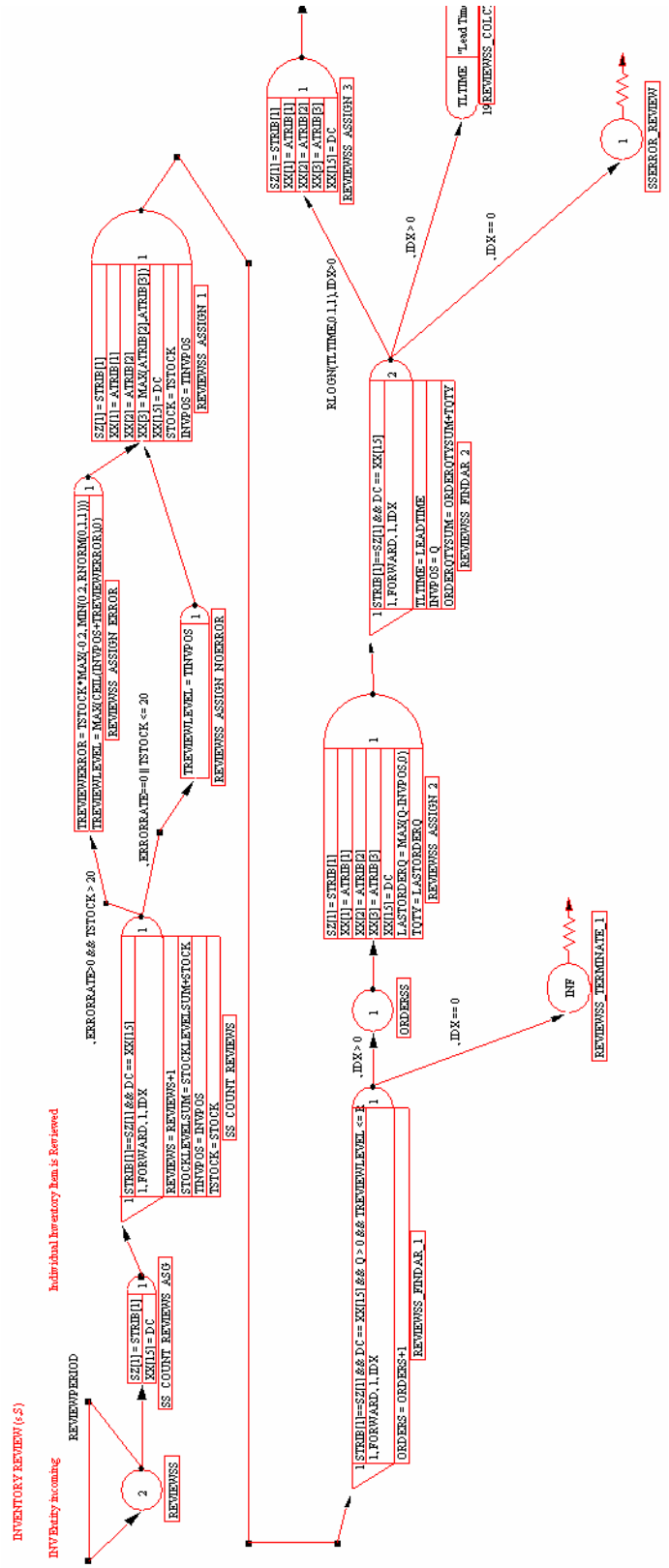

TIV= TIV + MAX(UPRICE*STOCK,0)
LDINV_ASSIGN_1
,DC == 1
,DC == 2
,DC == 3
TIDC1 = TIDC1 + MAX(UPRICE*STOCK,0)
TIDC2 = TIDC2 + MAX(UPRICE*STOCK,0)
LDINV2_ASSIGN_9
TIDC3 = TIDC3 + MAX(UPRICE*STOCK,0)
LDINV2_ASSIGN_7
LDINV2_GOON_1
,CD == 0
,CD == 1
TIVCD = TIVCD + MAX(UPRICE*STOCK,0)
LDINV_ASSIGN_3
LDINVSP_GOON_1
UNFRM(0,(REVIEWPERIOD-1)),POLICY <= 4
"REVIEWRQ"
UNFRM(0,(REVIEWPERIOD-1)),POLICY >= 5
"REVIEWSS"
PolicyError
INF
,DC < 1 || DC > 3
LDINV2_EVENT_1
ERROR_wrong_DC

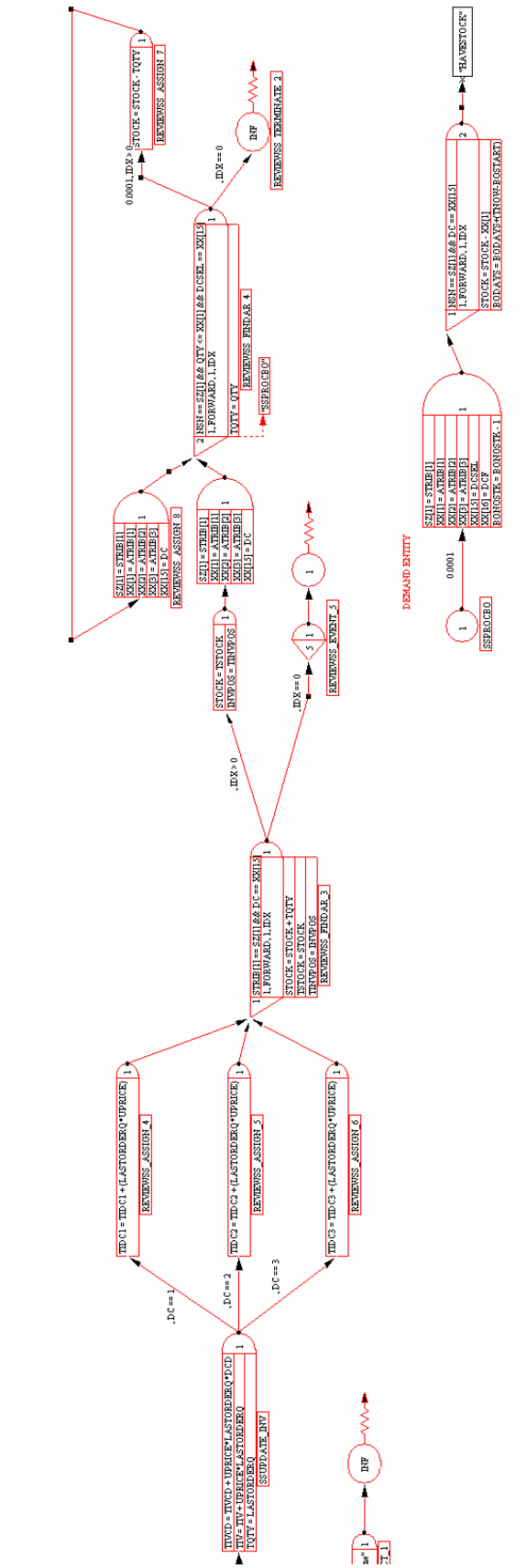

RECS FOR INVENTORY
INF
0.001
IDXREC = 1
CASER
NSN
QTY
DATE
DELTIME
SHIPTIME
IPG
DUPRICE
DC1
F1
DC2
F2
DC3
F3
RECS_READ_1
,IDX > 0
,IDX == 0
RECS_GOON_1
DATE - TNOW, DATE == IDXREC
1, DATE > IDXREC
TRR = TRR + 1
RECS_ASSIGN_1
IDXREC = IDXREC + 1
RECS_ASSIGN_2
INF
RECS_TERMINATE_1

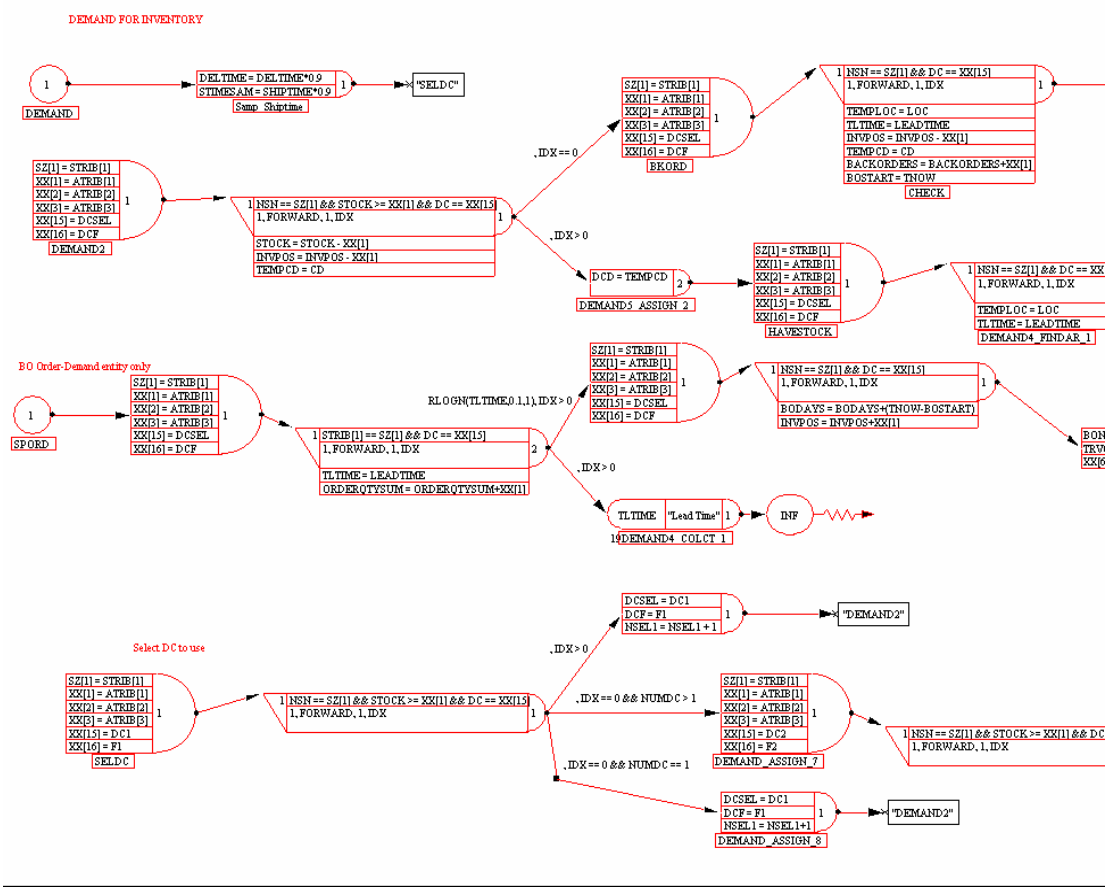

SZ[1] = STRIB[1]
1
1 STRIB[1] == SZ[1]
1, FORWARD, 1, IDX3-1
,IDX3 >0
1
"Demand"
,IDX3 == 0
3
1
ERROR
1
ERROR_No_Inv_Rec
1
,IPG == 1
,IPG == 2
,IPG == 3
TRR1 = TRR1 + 1
1
TRR2 = TRR2 + 1
1
RECS4_ASSIGN_3
TRR3 = TRR3 + 1
1
RECS4_ASSIGN_4
1
RECS4_GOON_2

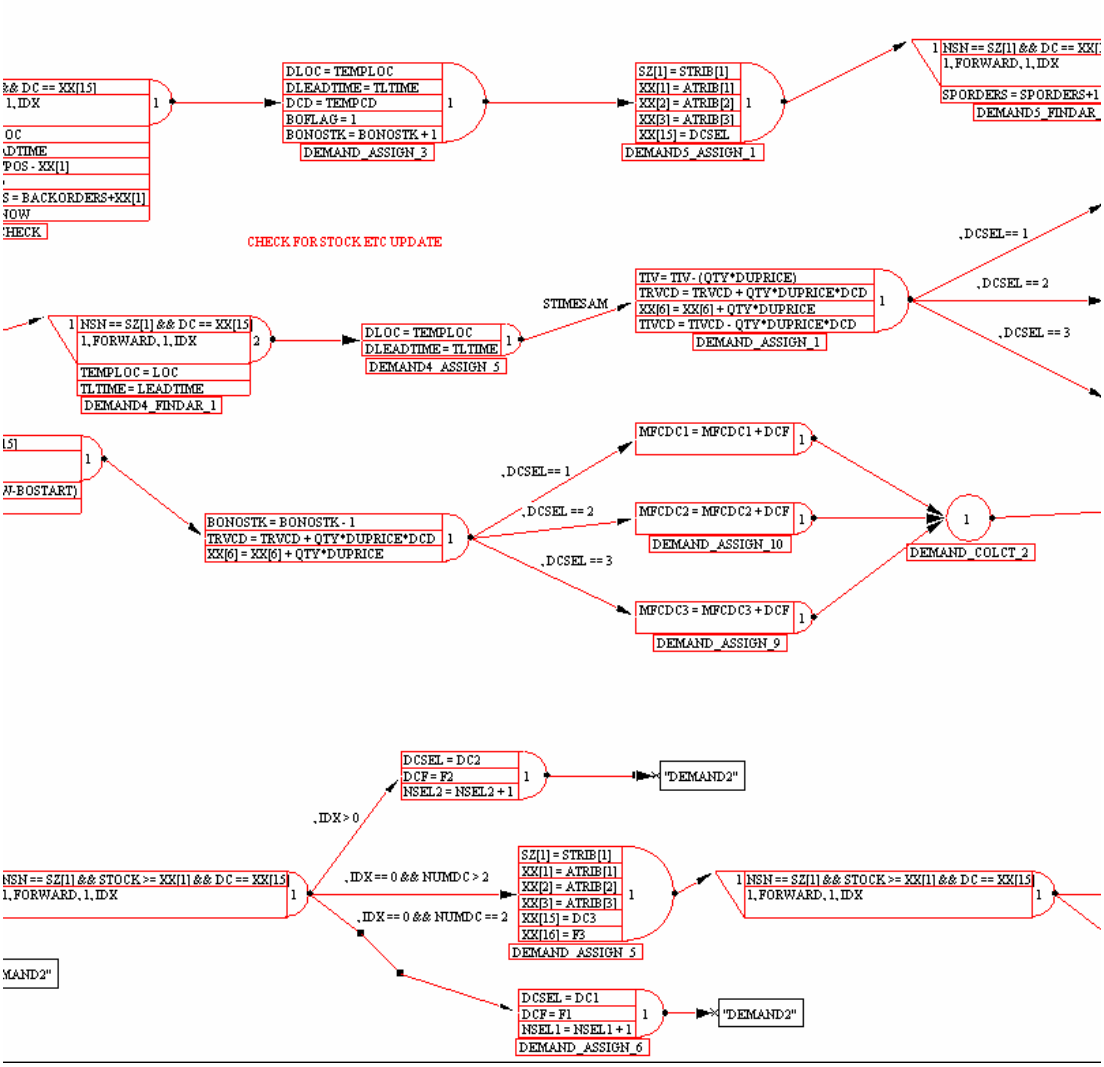

INVENTORY REVIEW (r, Q)
Individual Inventory Item is Reviewed
INVEntity incoming
REVIEWPERIOD
2
REVIEWRQ
SZ[1] = STRIB[1]
XX[15] = DC
1
COUNT_REVIEWS_ASG
1 STRIB[1]==SZ[1] && DC == XX[15]
1, FORWARD, 1, IDX
REVIEWS = REVIEWS+1
STOCKLEVELSUM = STOCKLEVELSUM+STOCK
TINVPOS = INVPOS
TSTOCK = STOCK
COUNT_REVIEWS
, ERRORRATE>0 && TSTOCK>20
TREVIEWERROR = TSTOCK*MAX(-0.2, MIN(0.2, RNORM(0,1,1)))
TREVIEWLEVEL = MAX(CEIL(TINVPOS+TREVIEWERROR),0)
REVIEWRQ_ASSIGN_ERROR
, ERRORRATE==0 || TSTOCK <= 20
TREVIEWLEVEL = TINVPOS
REVIEWRQ_ASSIGN_NOERROR
SZ[1] = STRIB[1]
XX[1] = ATRIB[1]
XX[2] = ATRIB[2]
XX[3] = ATRIB[3]
XX[15] = DC
STOCK = TSTOCK
INVPOS = TINVPOS
REVIEW4_ASSIGN_8
1 STRIB[1]==SZ[1] && DC == XX[15] && Q > 0 && TREVIEWLEVEL <= R
1, FORWARD, 1, IDX
ORDERS = ORDERS+1
REVIEW4_FINDAR_2
, IDX > 0
ORDER
, IDX == 0
INF
REVIEW_TERMINATE_1
SZ[1] = STRIB[1]
XX[1] = ATRIB[1]
XX[2] = ATRIB[2]
XX[3] = ATRIB[3]
XX[15] = DC
REVIEW4_ASSIGN_4
1 STRIB[1]==SZ[1] && DC == XX[15]
1, FORWARD, 1, IDX
TLTIME = LEADTIME
INVPOS = INVPOS + Q
ORDERQTYSUM = ORDERQTYSUM+Q
REVIEW4_FINDAR_1
RLOGN(TLTIME,0.1,1), IDX>0
SZ[1] = STRIB[1]
XX[1] = ATRIB[1]
XX[2] = ATRIB[2]
XX[3] = ATRIB[3]
XX[15] = DC
REVIEW4_ASSIGN_2
TIVCD = TIVCD + UPRICE*Q*DCD
TIV = TIV + UPRICE*Q
UPDATE_INV
, IDX > 0
TLTIME "Lead Time"
19 REVIEW4_COLCT_1
INF
, IDX == 0
ERROR_REVIEW

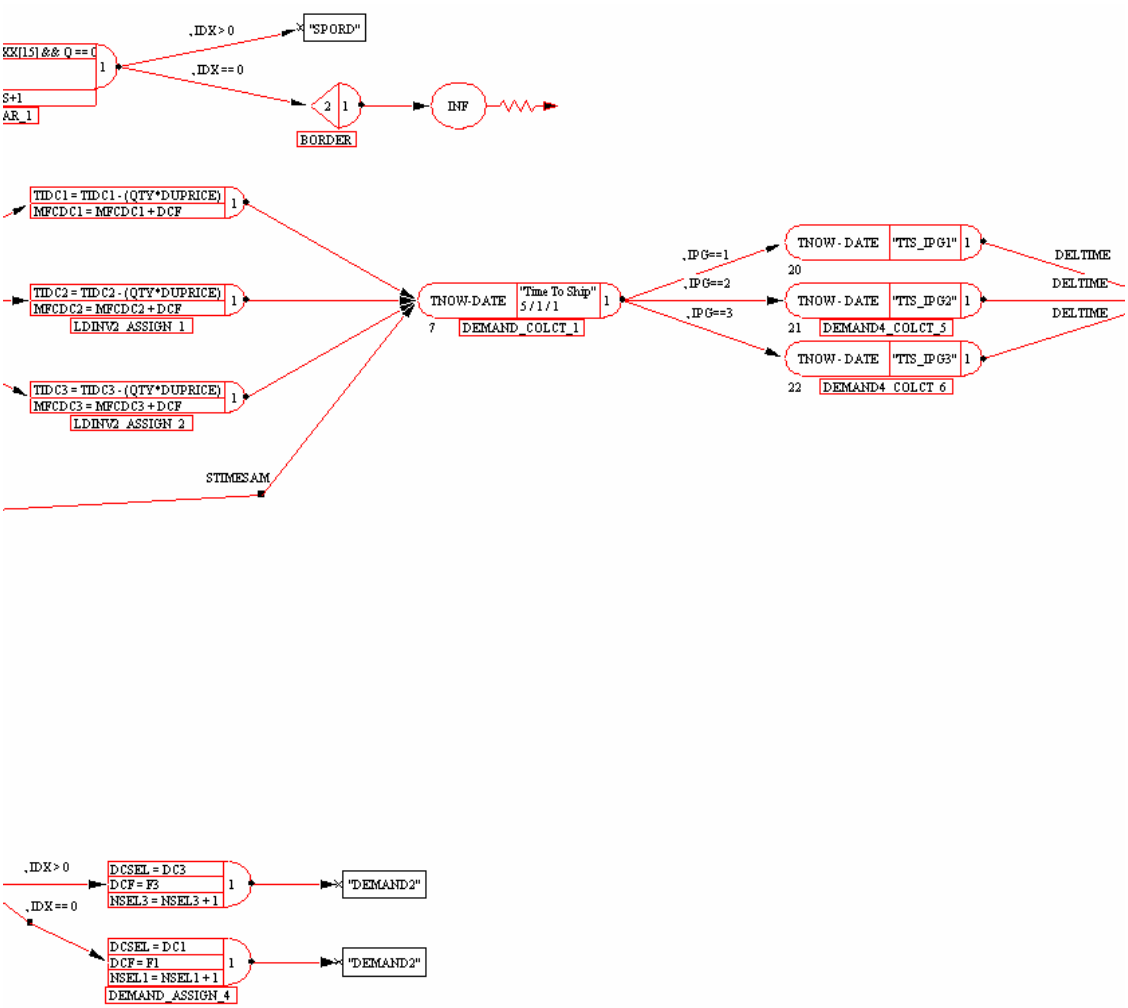

"LAVESTOCK"
STOCK = STOCK - XX[1]
BODAYS = BODAYS+TNOW-BOSTART
1 NSN == SZ[1] && DC == XX[15]
1.FORWARD,.1.IDX
1
SZ[1] = STRIB[1]
XX[1] = ATRIB[1]
XX[2] = ATRIB[2]
XX[3] = ATRIB[3]
XX[15] = DCSEL
XX[16] = DCF
BONOSTK = BONOSTK - 1
DEMAND ENTITY
0.0001
1
PROCBO
STOCK = STOCK - TQTY
REVIEW5 ASSIGN 2
0.0001.IDX > 0
.IDX == 0
INF
REVIEW5 TERMINATE 1
2 NSN == SZ[1] && QTY <= XX[1] && DCSEL == XX[15]
1.FORWARD,.1.IDX
TQTY = QTY
REVIEW5_FIND AR_1
"PROCBO"
SZ[1] = STRIB[1]
XX[1] = ATRIB[1]
XX[2] = ATRIB[2]
XX[3] = ATRIB[3]
XX[15] = DC
REVIEW5 ASSIGN 1
SZ[1] = STRIB[1]
XX[1] = ATRIB[1]
XX[2] = ATRIB[2]
XX[3] = ATRIB[3]
XX[15] = DC
STOCK = TSTOCK
INVPOS = TINVPOS
.IDX > 0
1
.IDX == 0
5
1
REVIEW4 EVENT 1
1
STRIB[1] == SZ[1] && DC == XX[15]
1.FORWARD,.1.IDX
STOCK = STOCK + Q
TSTOCK = STOCK
TINVPOS = INVPOS
REVIEW3_FIND AR_2
.DC == 1
.DC == 2
.DC == 3
TDC1 = TDC1 + (Q*UPRICE)
REVIEW4 ASSIGN 1
TDC2 = TDC2 + (Q*UPRICE)
NODE_9
TDC3 = TDC3 + (Q*UPRICE)
LDINV2 ASSIGN 8

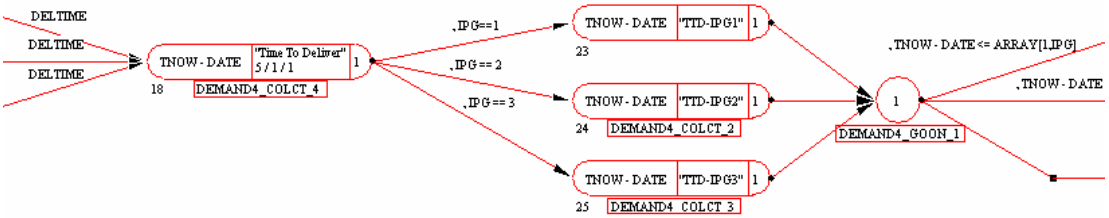

INVENTORY REVIEW (4.5)
INV Entity incoming
Individual Inventory Item is Reviewed
REVIEWPERIOD
REVIEWSS
SZ[1] = STRIB[1]
XX[15] = DC
SS_COUNT_REVIEWS_ASG
STRIB[1]=SZ[1] && DC == XX[15]
1,FORWARD,1,IDX
REVIEWS = REVIEWS+1
STOCKLEVELSUM = STOCKLEVELSUM+STOCK
INVPOS = INVPOS
TSTOCK = STOCK
SS_COUNT_REVIEWS
.ERRORRATE>0 && TSTOCK > 20
TREVIEWERROR = TSTOCK*MAX(.0,2,MIN(0.2,RNORM(0,1,1)))
TREVIEWLEVEL = MAX(CEIL(INVPOS+TREVIEWERROR),0)
REVIEWSS_ASSIGN_ERROR
.ERRORRATE==0 | TSTOCK <= 20
TREVIEWLEVEL = TINVPOS
REVIEWSS_ASSIGN_NOERROR
SZ[1] = STRIB[1]
XX[1] = ATRIB[1]
XX[2] = ATRIB[2]
XX[3] = MAX(ATRIB[2],ATRIB[3])
XX[15] = DC
STOCK = TSTOCK
INVPOS = TINVPOS
REVIEWSS_ASSIGN_1
STRIB[1]=SZ[1] && DC == XX[15] && Q > 0 && TREVIEWLEVEL <= I
1,FORWARD,1,IDX
ORDERS = ORDERS+1
REVIEWSS_FINDAR_1
.IDX > 0
ORDERSS
.IDX == 0
INF
REVIEWSS_TERMINATE_1
SZ[1] = STRIB[1]
XX[1] = ATRIB[1]
XX[2] = ATRIB[2]
XX[3] = ATRIB[3]
XX[15] = DC
LASTORDERQ = MAX(Q-INVPOS,0)
TQTY = LASTORDERQ
REVIEWSS_ASSIGN_2
STRIB[1]=SZ[1] && DC == XX[15]
1,FORWARD,1,IDX
TLTIME = LEADTIME
INVPOS = Q
ORDERQTYSUM = ORDERQTYSUM+TQTY
REVIEWSS_FINDAR_2
RLOGN(TLTIME,0.1,1),IDX>0
.IDX > 0
.IDX == 0
SZ[1] = STRIB[1]
XX[1] = ATRIB[1]
XX[2] = ATRIB[2]
XX[3] = ATRIB[3]
XX[15] = DC
REVIEWSS_ASSIGN_3
TLTIME
"Lead Tim
19REVIEWSS_COLC
SSERROR_REVIEW

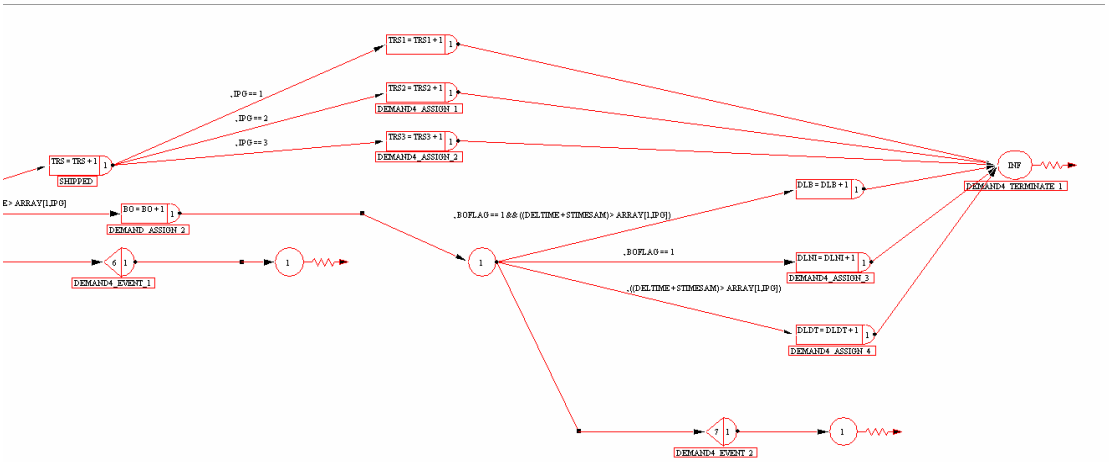

DEMAND ENTITY
SSPROCBO
0.0001
SZ[1] = STRIB[1]
XX[1] = ATRIB[1]
XX[2] = ATRIB[2]
XX[3] = ATRIB[3]
XX[15] = DCSEL
XX[16] = DCF
EONOSTK = EONOSTK - 1
1 NSN == SZ[1] && DC == XX[15] 1.FORWARD, 1.IDX
STOCK = STOCK - XX[1]
EODAYS = EODAYS-TNOW-BOSTART
"HAVESTOCK"
SZ[1] = STRIB[1]
XX[1] = ATRIB[1]
XX[2] = ATRIB[2]
XX[3] = ATRIB[3]
XX[15] = DC
REVDEWSS_ASSIGN_8
2 NSN == SZ[1] && QTY <= XX[1] && DCSEL == XX[15] 1.FORWARD, 1.IDX
TQTY = QTY
REVDEWSS_FINDJAR_4
"SSPROCBO"
.IDX== 0
DNF
REVDEWSS_TERMINATE_2
0.0001, IDX> 0
STOCK = STOCK - TQTY
REVDEWSS_ASSIGN_7
STOCK = TSTOCK
INVPOS = TINVPOS
.IDX> 0
.IDX== 0
REVDEWSS_EVENT_5
STRIB[1] == SZ[1] && DC == XX[15] 1.FORWARD, 1.IDX
STOCK = STOCK + TQTY
TSTOCK = STOCK
TINVPOS = INVPOS
REVDEWSS_FINDJAR_3
TTD C1 = TTD C1 + (LASTORDERQ*UPRICE)
REVDEWSS_ASSIGN_4
TTD C2 = TTD C2 + (LASTORDERQ*UPRICE)
REVDEWSS_ASSIGN_5
TTD C3 = TTD C3 + (LASTORDERQ*UPRICE)
REVDEWSS_ASSIGN_6
.DC == 1
.DC == 2
.DC == 3
TTVCD = TTVCD + UPRICE*LASTORDERQ*DCD
TTQ = TTQ + LASTORDERQ
TQTY = LASTORDERQ
SSUPDATE_INV
DNF

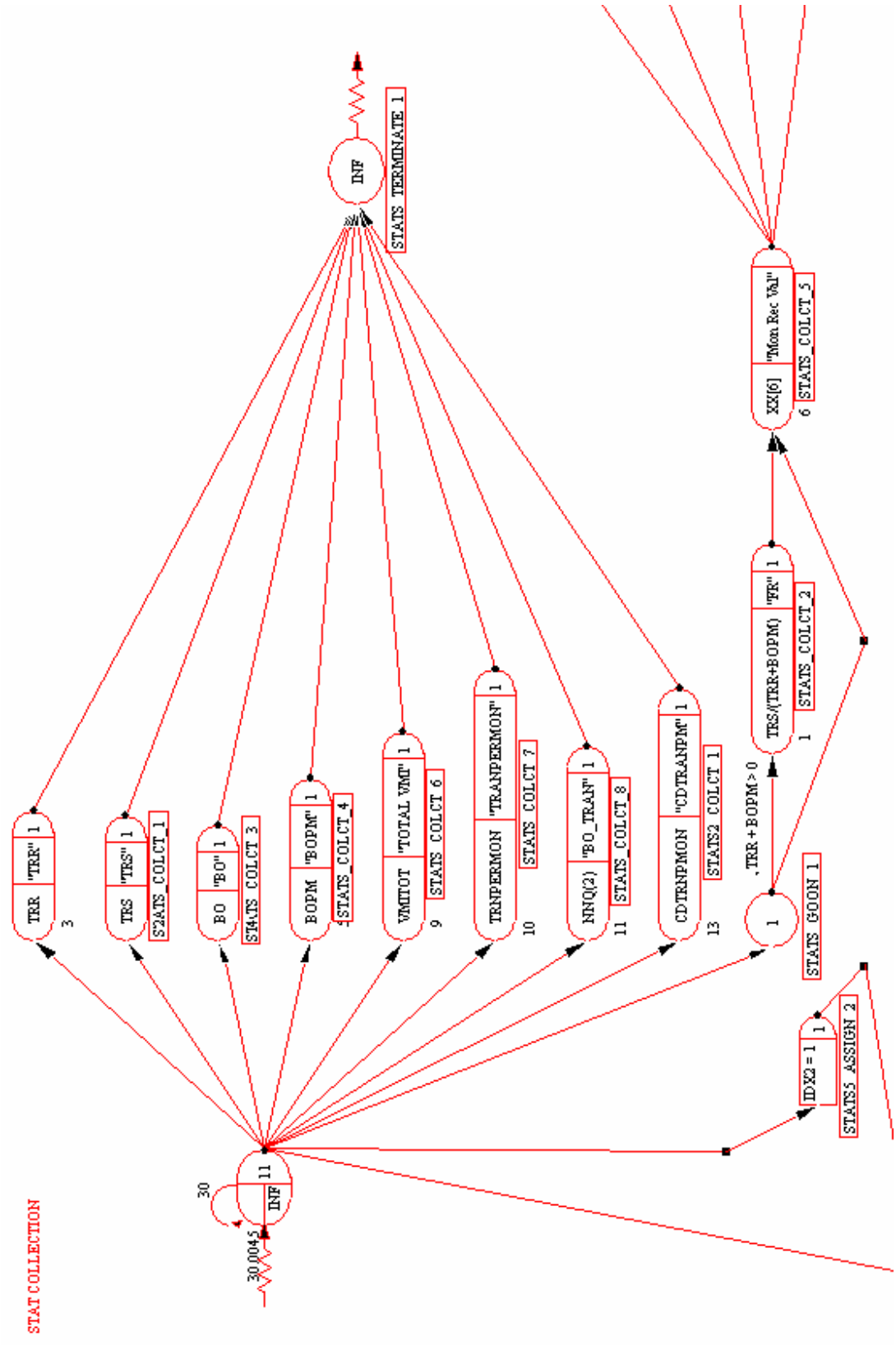

DEMAND FOR INVENTORY
DEMAND
DELTIME = DELTIME*0.9
STIMESAM = SHIPTIME*0.9
Samp Shiptime
"SELDC"
SZ[1] = STRIB[1]
XX[1] = ATRIB[1]
XX[2] = ATRIB[2]
XX[3] = ATRIB[3]
XX[15] = DCSEL
XX[16] = DCF
DEMAND2
NSN == SZ[1] && STOCK >= XX[1] && DC == XX[15]
1, FORWARD, 1, IDX
STOCK = STOCK - XX[1]
INVPOS = INVPOS - XX[1]
TEMPCD = CD
.IDX == 0
.IDX > 0
SZ[1] = STRIB[1]
XX[1] = ATRIB[1]
XX[2] = ATRIB[2]
XX[3] = ATRIB[3]
XX[15] = DCSEL
XX[16] = DCF
BKORD
NSN == SZ[1] && DC == XX[15]
1, FORWARD, 1, IDX
TEMPLOC = LOC
TLTIME = LEADTIME
INVPOS = INVPOS - XX[1]
TEMPCD = CD
BACKORDERS = BACKORDERS+XX[1]
BOSTART = TNOW
CHECK
DCD = TEMPCD
DEMAND5_ASSIGN_2
SZ[1] = STRIB[1]
XX[1] = ATRIB[1]
XX[2] = ATRIB[2]
XX[3] = ATRIB[3]
XX[15] = DCSEL
XX[16] = DCF
HAVESTOCK
NSN == SZ[1] && DC == XX
1, FORWARD, 1, IDX
TEMPLOC = LOC
TLTIME = LEADTIME
DEMAND4_FINDAR_1
BO Order-Demand entity only
SPORD
SZ[1] = STRIB[1]
XX[1] = ATRIB[1]
XX[2] = ATRIB[2]
XX[3] = ATRIB[3]
XX[15] = DCSEL
XX[16] = DCF
STRIB[1] == SZ[1] && DC == XX[15]
1, FORWARD, 1, IDX
TLTIME = LEADTIME
ORDERQTYSUM = ORDERQTYSUM+XX[1]
RLOGN(TLTIME,0.1,1),IDX > 0
.IDX > 0
SZ[1] = STRIB[1]
XX[1] = ATRIB[1]
XX[2] = ATRIB[2]
XX[3] = ATRIB[3]
XX[15] = DCSEL
XX[16] = DCF
NSN == SZ[1] && DC == XX[15]
1, FORWARD, 1, IDX
BODAYS = BODAYS+(TNOW-BOSTART)
INVPOS = INVPOS+XX[1]
TLTIME "Lead Time"
INF
DEMAND4_COLCT_1
Select DC to use
SZ[1] = STRIB[1]
XX[1] = ATRIB[1]
XX[2] = ATRIB[2]
XX[3] = ATRIB[3]
XX[15] = DC1
XX[16] = F1
SELDC
NSN == SZ[1] && STOCK >= XX[1] && DC == XX[15]
1, FORWARD, 1, IDX
.IDX > 0
.IDX == 0 && NUMDC > 1
.IDX == 0 && NUMDC == 1
DCSEL = DC1
DCF = F1
NSEL1 = NSEL1 + 1
"DEMAND2"
SZ[1] = STRIB[1]
XX[1] = ATRIB[1]
XX[2] = ATRIB[2]
XX[3] = ATRIB[3]
XX[15] = DC2
XX[16] = F2
DEMAND_ASSIGN_7
NSN == SZ[1] && STOCK >= XX[1] && DC
1, FORWARD, 1, IDX
DCSEL = DC1
DCF = F1
NSEL1 = NSEL1+1
DEMAND_ASSIGN_8
"DEMAND2"

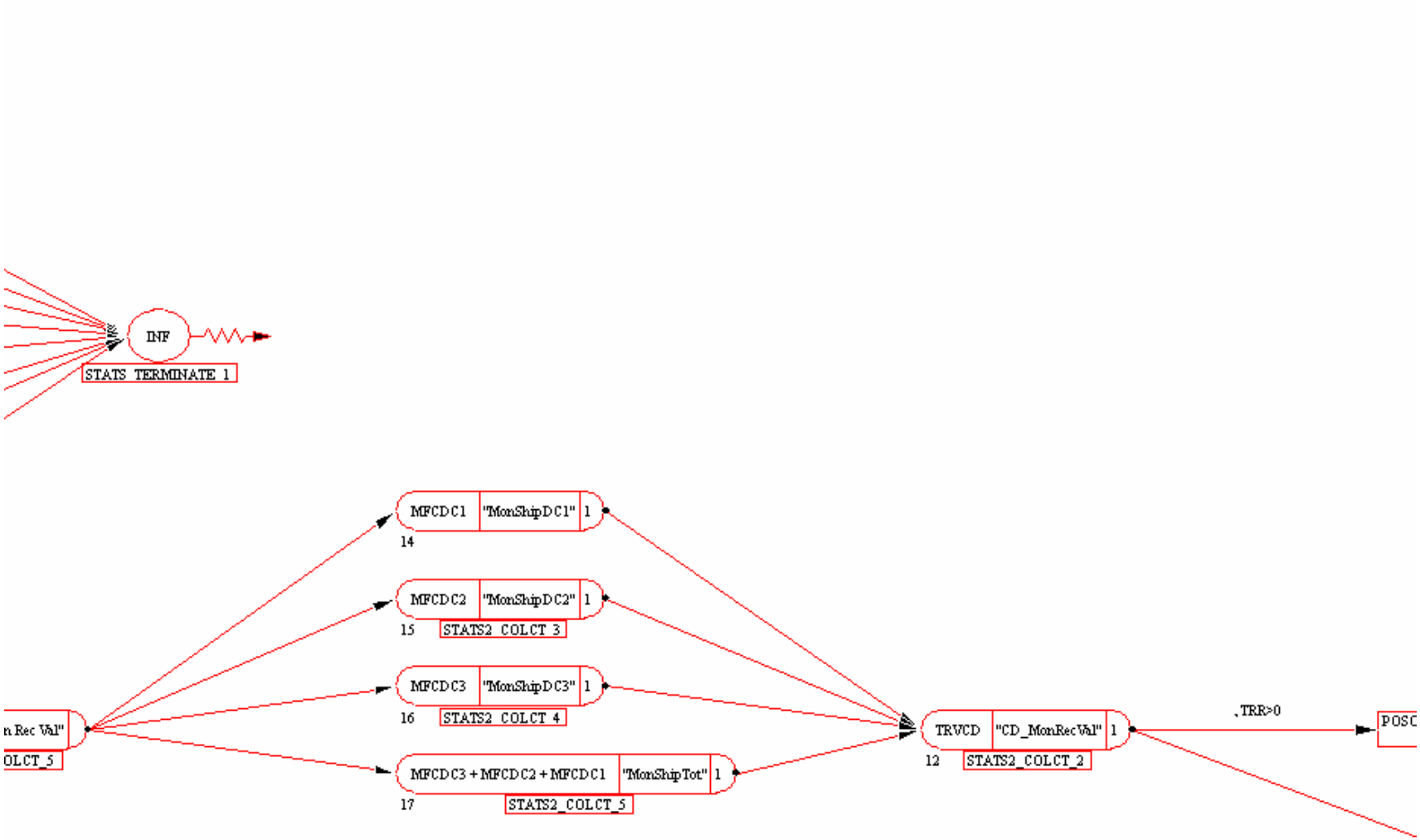

DLOC = TEMPLOC
DLEADTIME = TLTIME
DCD = TEMPCD
BOFLAG = 1
BONOSTK = BONOSTK + 1
DEMAND_ASSIGN_3
SZ[1] = STRIB[1]
XX[1] = ATRIB[1]
XX[2] = ATRIB[2]
XX[3] = ATRIB[3]
XX[15] = DCSEL
DEMAND5_ASSIGN_1
1,FORWARD,1,IDX
SPORDERS = SPORDERS+1
DEMAND5_FINDAR_
CHECK FOR STOCK ETC UPDATE
1,FORWARD,1,IDX
TEMPLOC = LOC
TLTIME = LEADTIME
DEMAND4_FINDAR_1
DLOC = TEMPLOC
DLEADTIME = TLTIME
DEMAND4_ASSIGN_5
STIMESAM
TIV = TIV - (QTY*DUPRICE)
TRVCD = TRVCD + QTY*DUPRICE*DCD
XX[6] = XX[6] + QTY*DUPRICE
TIVCD = TIVCD - QTY*DUPRICE*DCD
DEMAND_ASSIGN_1
,DCSEL== 1
,DCSEL == 2
,DCSEL == 3
BONOSTK = BONOSTK - 1
TRVCD = TRVCD + QTY*DUPRICE*DCD
XX[6] = XX[6] + QTY*DUPRICE
MFCDC1 = MFCDC1 + DCF
MFCDC2 = MFCDC2 + DCF
DEMAND_ASSIGN_10
MFCDC3 = MFCDC3 + DCF
DEMAND_ASSIGN_9
DEMAND_COLCT_2
DCSEL = DC2
DCF = F2
NSEL2 = NSEL2 + 1
"DEMAND2"
,IDX > 0
,IDX == 0 && NUMDC > 2
,IDX == 0 && NUMDC == 2
1,FORWARD,1,IDX
SZ[1] = STRIB[1]
XX[1] = ATRIB[1]
XX[2] = ATRIB[2]
XX[3] = ATRIB[3]
XX[15] = DC3
XX[16] = F3
DEMAND_ASSIGN_5
DCSEL = DC1
DCF = F1
NSEL1 = NSEL1 + 1
DEMAND_ASSIGN_6


,IDX > 0
"SPORD"
,IDX == 0
BORDER
INF
TIDC1 = TIDC1 - (QTY*DUPRICE)
MFCDC1 = MFCDC1 + DCF
TIDC2 = TIDC2 - (QTY*DUPRICE)
MFCDC2 = MFCDC2 + DCF
LDINV2_ASSIGN_1
TIDC3 = TIDC3 - (QTY*DUPRICE)
MFCDC3 = MFCDC3 + DCF
LDINV2_ASSIGN_2
STIMESAM
TNOW-DATE
"Time To Ship"
5/1/1
DEMAND_COLCT_1
,IPG==1
,IPG==2
,IPG==3
TNOW-DATE
"TTS_IPG1"
20
TNOW-DATE
"TTS_IPG2"
21
DEMAND4_COLCT_5
TNOW-DATE
"TTS_IPG3"
22
DEMAND4_COLCT_6
DELTIME
DELTIME
DELTIME
,IDX > 0
,IDX == 0
DCSEL = DC3
DCF = F3
NSEL3 = NSEL3 + 1
"DEMAND2"
DCSEL = DC1
DCF = F1
NSEL1 = NSEL1 + 1
DEMAND_ASSIGN_4
"DEMAND2"

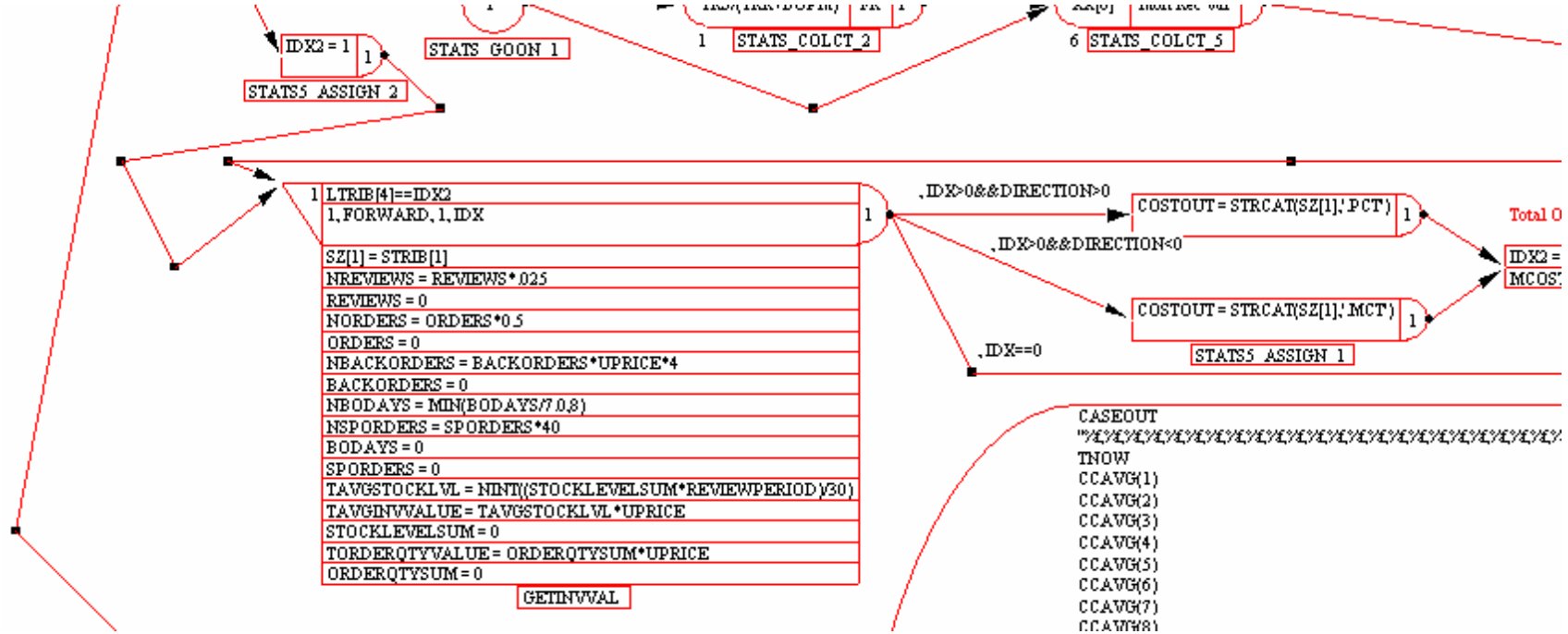

DELTIME
DELTIME
DELTIME
TNOW - DATE
"Time To Deliver"
5 / 1 / 1
1
18
DEMAND4_COLCT_4
,IPG==1
,IPG== 2
,IPG== 3
TNOW - DATE
"TTD-IPG1"
1
23
TNOW - DATE
"TTD-IPG2"
1
24
DEMAND4_COLCT_2
TNOW - DATE
"TTD-IPG3"
1
25
DEMAND4_COLCT_3
1
DEMAND4_GOON_1
,TNOW - DATE<= ARRAY[1,IPG]
,TNOW - DATE

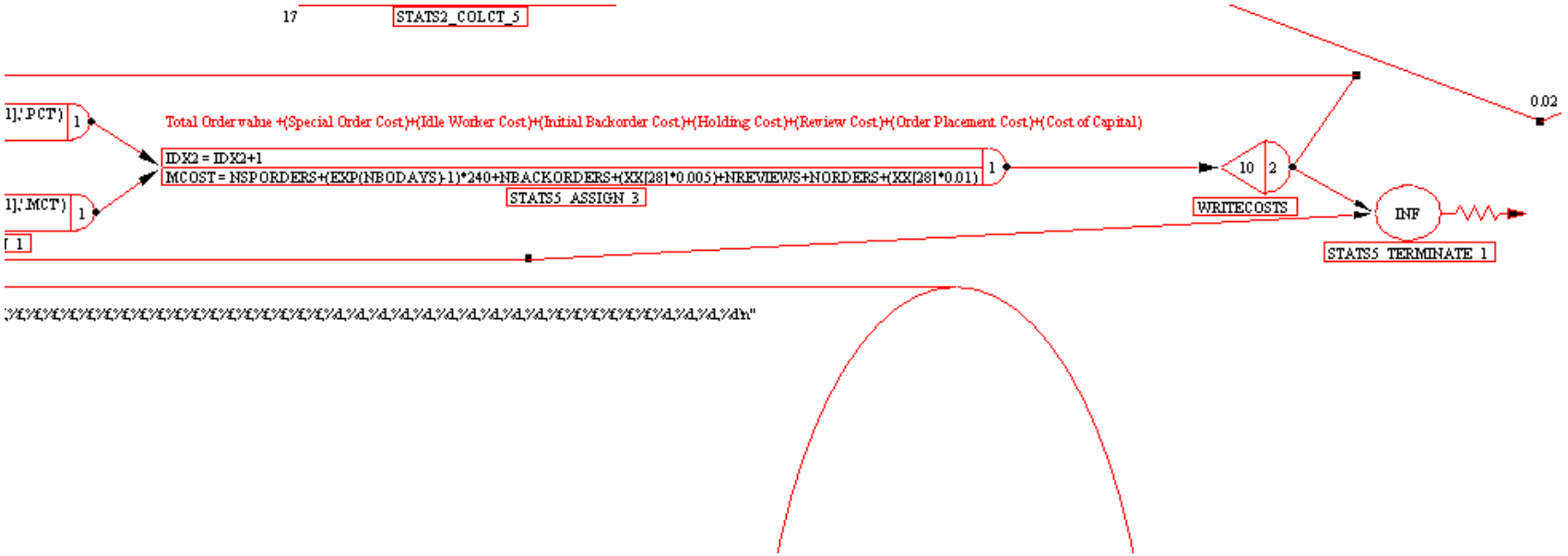

TRS1 = TRS1 + 1
TRS2 = TRS2 + 1
DEMAND4_ASSIGN_1
TRS3 = TRS3 + 1
DEMAND4_ASSIGN_2
,IPG== 1
,IPG== 2
,IPG== 3
TRS = TRS + 1
SHIPPED
INF
DEMAND4_TERMINATE_1
E> ARRAY[1,IPG]
BO = BO + 1
DEMAND_ASSIGN_2
DEMAND4_EVENT_1
,BOFLAG== 1 && ((DELTIME+STIMESAM)> ARRAY[1,IPG])
DLB = DLB + 1
,BOFLAG== 1
DLNI = DLNI + 1
DEMAND4_ASSIGN_3
,((DELTIME+STIMESAM)> ARRAY[1,IPG])
DLDT = DLDT + 1
DEMAND4_ASSIGN_4
DEMAND4_EVENT_2


STAT COLLECTION
TRR "TRR" 1
3
TRS "TRS" 1
S2ATS_COLCT_1
BO "BO" 1
ST4TS_COLCT_3
BOPM "BOPM" 1
STATS_COLCT_4
VMITOT "TOTAL VMI" 1
9 STATS_COLCT_6
TRNPERMON "TRANPERMON" 1
10 STATS_COLCT_7
NNQ(2) "BO_TRAN" 1
11 STATS_COLCT_8
CDTRNPMON "CDTRANPM" 1
13 STATS2_COLCT_1
INF
STATS_TERMINATE 1
30
11
INF
30.0045
1
.TRR + BOPM > 0
TRS/(TRR+BOPM) "FR" 1
1 STATS_COLCT_2
XX[6] "Mon Rec Val"
6 STATS_COLCT_5
IDX2 = 1 1
STATS_ASSIGN_2
STATS_GOON_1


INF
STATS_TERMINATE 1
MFCDC1 "MonShipDC1" 1
14
MFCDC2 "MonShipDC2" 1
15 STATS2_COLCT_3
MFCDC3 "MonShipDC3" 1
16 STATS2_COLCT_4
MFCDC3+MFCDC2+MFCDC1 "MonShipTot" 1
17 STATS2_COLCT_5
TRVCD "CD_MonRecVal" 1
12 STATS2_COLCT_2
,TRR>0
POSC

INF
XX[6] = 0
TRNPERMON = 0
TRVCD = 0
CDTRNPMON = 0
MFCDC3 = 0
MFCDC2 = 0
MFCDC1 = 0
NSEL1 = 0
NSEL2 = 0
NSEL3 = 0
1
TRR = 0
TRS = 0
BOPM = BO
BO = 0
TRR1 = 0
TRR2 = 0
TRR3 = 0
TRS1 = 0
TRS2 = 0
TRS3 = 0
DLNT = 0
DLDT = 0
DLB = 0
GBL = 0
NORDERS = 0
NREVIEWS = 0
STATS ASSIGN 1
1
.02
POSOT
"Perc Ord Ship OnTime"
1
8
POSOT = 1-BO/TRR
1
0.02
10
2


IDX2 = 1
1
STATS5_ASSIGN_2
STATS_GOON_1
1
STATS_COLCT_2
6
STATS_COLCT_5
1
LTRIB[4]==IDX2
1,FORWARD,1,IDX
SZ[1] = STRIB[1]
NREVIEWS = REVIEWS*.025
REVIEWS = 0
NORDERS = ORDERS*0.5
ORDERS = 0
NBACKORDERS = BACKORDERS*UPRICE*4
BACKORDERS = 0
NBODAYS = MIN(BODAYS/7.0,8)
NSPORDERS = SPORDERS*40
BODAYS = 0
SPORDERS = 0
TAVGSTOCKLVL = NINT((STOCKLEVELSUM*REVIEWPERIOD)/30)
TAVGINVVALUE = TAVGSTOCKLVL*UPRICE
STOCKLEVELSUM = 0
TORDERQTYVALUE = ORDERQTYSUM*UPRICE
ORDERQTYSUM = 0
GETINVVAL
1
,IDX>0&&DIRECTION>0
COSTOUT = STRCAT(SZ[1],'PCT')
1
,IDX>0&&DIRECTION<0
COSTOUT = STRCAT(SZ[1],'MCT')
1
STATS5_ASSIGN_1
,IDX==0
Total O
IDX2 =
MCOST
CASEOUT
TNOW
CCAVG(1)
CCAVG(2)
CCAVG(3)
CCAVG(4)
CCAVG(5)
CCAVG(6)
CCAVG(7)


17
STATS2_COLCT_5
0.02
Total Order value +(Special Order Cost)+(Idle Worker Cost)+(Initial Backorder Cost)+(Holding Cost)+(Review Cost)+(Order Placement Cost)+(Cost of Capital)
IDX2 = IDX2+1
MCOST = NSPORDERS+(EXP(NBODAYS)-1)*240+NBACKORDERS+(XX[28]*0.005)+NREVIEWS+NORDERS+(XX[28]*0.01)
STATS5_ASSIGN_3
10
2
WRITECOSTS
INF
STATS5_TERMINATE_1

Appendix D

SPSA simulation optimization implementation c source code

```
// spsa.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"


	int const MAXINV = 2000;

static double readCost(char * filename, int plus)
{
	FILE *fpeab;
	char parmfile[80] = "PARAMS.PRM";
	errno_t err;
	float cost[MAXINV], tcost=0;
	int i;

	err = strcpy_s(parmfile, _countof(parmfile),filename);
	if (plus == 1)
		err = strcat_s(parmfile, _countof(parmfile),".PCT");
	else
		err = strcat_s(parmfile, _countof(parmfile),".MCT");
	err = fopen_s(&fpeab,parmfile,"r");
	if (err != 0) {
		printf("read costfile error");
			return 0;
	} else {
		for(i=0;fscanf_s(fpeab,"%f",&cost[i])>0;i++)
			tcost = tcost+cost[i];
		fclose(fpeab);
		err = remove(parmfile);
		if (i > 0) {
			return tcost/i;
//			return tcost;
		}
		else {
			return 0;
		}
	}
}

static double readCostNoise(char * filename, int periods, int opt, double * meanCost_out)
{
	FILE *fpeab;
	char parmfile[80] = "PARAMS.PRM";
	errno_t err;
	double cost[MAXINV], tcost=0.0, costs[100], costMean, sumOfSquares = 0.0, noise;
	int i,samples = 0, period = 1;

	err = strcpy_s(parmfile, _countof(parmfile),filename);
	if (opt) {
		err = strcat_s(parmfile, _countof(parmfile),"OPT.CST");
	} else {
		err = strcat_s(parmfile, _countof(parmfile),".CST");
	}
	err = fopen_s(&fpeab,parmfile,"r");
	if (err != 0) {
		printf("read costfile error");
			return 0;
```

```
	} else {
		for(i=0;fscanf_s(fpeab,"%lf",&cost[i])>0;i++) {
			tcost = tcost+cost[i];
			if (period==periods) {
				costs[samples++] = tcost/periods;
				tcost = 0.0;
				period = 0;
			}
			period++;
		}
		fclose(fpeab);
		err = remove(parmfile);
		costMean = 0.0;
		sumOfSquares = 0.0;
		for (i=0;i<samples;i++) {
			costMean = costMean+costs[i];
		}
		costMean = costMean/(double) samples;
		*meanCost_out = costMean;
		for (i=0;i<samples;i++) {
			sumOfSquares = sumOfSquares+pow((costs[i]-costMean),2.0);
		}
		if (samples > 1)
			noise = sqrt((sumOfSquares/(double)(samples - 1)));
		if ( samples > 1) {
			return noise;
		}
		else {
			return 0;
		}
	}
}


int main(int argc, char * argv[])
{
	FILE *fpeab;
	char SKU[MAXINV][20];
	FILE *fpeab1;
	int runs = 7, minsFound = 0, parameters = 2;
	char parmfile[80] = "PARAMS.PRM";
	char inventory[80] = "INVENTORY.TXT";
	char optout[80] =  "OPT";
	char scenario[80] = "TESTING1";
	struct testDiff {
		int lastDiffIdx;
		int minArgCount;
		double avgDiff;
		double madDiff;
		double min;
		double minArgs[3];
		double movingAvg;
		double costAvgs[10];
		double costDiff[10];
	};

	struct testDiff td[MAXINV];

	char command[80] = "C:/awesim/bin/execute.exe ";
	int controlPolicy = 0,  itemid[MAXINV], level[MAXINV],  SCL[MAXINV], RP[MAXINV], LT[MAXINV],
minFound[MAXINV], minCounted[MAXINV],
		i,j,k, m, LOC[MAXINV], CD[MAXINV], DC[MAXINV], SKUs, loopControl, A = 100,
reviewperiod[MAXINV], policy[MAXINV], demandtype[MAXINV];
	double  itemcost[MAXINV], deltas[3], ghats[3],  d_SCL[MAXINV],d_RP[MAXINV], d_reviewPeriod[MAXINV],
d_dailydemand[MAXINV], d_sd_demand[MAXINV],
		Ck[MAXINV], Ak[MAXINV][3], c[MAXINV], a[MAXINV][3], gamma = 0.101, alpha = 0.602,
pluscost[MAXINV], minuscost[MAXINV], d_orderlimit[MAXINV],
		d_errorrate[MAXINV], c_default = 10.0, a_default = 1.16, gradient, meanCost;
	errno_t err;
```

```
	//system("C:/awesim/bin/execute.exe MSIM");
	/*
	usage: spsa <scenario> <runs>
	   N = runs
		LOOP 1
			FOREACH inventory record
				calculate deltas, Ck, Ak
				write initial parameters (SCL, RP, deltas, Ck, Ak)
			simulate
		LOOP 2..N
			FOREACH inventory record
				read pluscost records (get avg)
				create new pluscost file
				read minuscost recors (get avg)
				create new minuscost file
				calulate ghats = avg(pluscost)-avg(minuscost)/(2*Ck*deltas)
				calculate new SCL, RP
				calculate deltas, Ck, Ak
				write parameters (SCL, RP, deltas, Ck, Ak)
			simulate

	*/

	switch (argc) {
	case 1:
	case 2:
	case 3:
		printf("usage: spsa <scenario> <inventoryfile> <runs> [ <control policy> <A> <gamma> <alpha>\n");
		printf(" <scenario> = The AWESIM scenario from the SPSAEXP project \n");
		printf(" <inventoryfile> = The inventory files is pairs. Check the SPSA control file for the name.\n");
		printf(" <runs> = iterations of the SPSA algorithm\n");
		printf(" <control policy> = The inventory control policy applied; if set > 0, overrides inventory file
input.\n");
		printf(" 1 = r, Q, and review period P are in the theta parameter vector.\n");
		printf(" 2 = r and Q are in the parameter set and the review period, P is read from the inventory file.\n");
		printf(" 3 = r is set to Q, Q and P are in the parameter set.\n");
		printf(" 4 = r is set to 0.5*Q and Q and P are in the parameter set.\n");
		printf(" 5 = s, S and P are in the parameter set.\n");
		printf(" 6 = s and S are in the parameter set and P is read from the inventory file.\n");
		printf(" 7 = S and P are in the parameter set and s = S.\n");
		printf(" 8 = S is in the parameter set, s = S-1 and P = 1.\n");
		printf(" <A> = Maximum iterations expected to convergence.\n");
		printf(" <gamma>  <alpha> are the gain sequence parameters for Ck and Ak respectively.\n");
		return -1;
		break;
	case 4:
		err = strcpy_s(scenario, _countof(scenario),argv[1]);
		err = strcpy_s(inventory, _countof(inventory),argv[2]);
		runs = atoi(argv[3]);
		break;
	case 5:
		err = strcpy_s(scenario, _countof(scenario),argv[1]);
		err = strcpy_s(inventory, _countof(inventory),argv[2]);
		runs = atoi(argv[3]);
		controlPolicy = atoi(argv[4]);
		break;
	case 6:
		err = strcpy_s(scenario, _countof(scenario),argv[1]);
		err = strcpy_s(inventory, _countof(inventory),argv[2]);
		runs = atoi(argv[3]);
		controlPolicy = atoi(argv[4]);
		A = atoi(argv[5]);
		break;
	case 7:
		err = strcpy_s(scenario, _countof(scenario),argv[1]);
		err = strcpy_s(inventory, _countof(inventory),argv[2]);
		runs = atoi(argv[3]);
		controlPolicy = atoi(argv[4]);
		A = atoi(argv[5]);
		gamma = atof(argv[6]);
```

```
                        break;
                default:
                        printf("usage: spsa <scenario> <inventoryfile> <runs> [ <control policy> <A> <gamma> <alpha>\n");
                        return -1;
                }

                if (parameters > 3) {
                        parameters = 3;
                }
                else if (parameters <= 0) {
                        parameters = 1;
                }
                for (i=0;i<3;i++) {
                        deltas[i] = (double) 0.0;
                }

/*
 Read the inventory file
*/
                err = fopen_s(&fpeab,inventory,"r");
                if (err != 0) {
                        printf("Can not open SKU file for reading");
                } else {
                        loopControl = 1;
                        for (i=0;loopControl > 0; i++) {
                                err = fscanf_s(fpeab,"%d %s %d %lf %lf %d %d %lf %d %d %d %d %lf %lf %d
%lf",&itemid[i], &SKU[i], 20, &level[i], &d_SCL[i], &d_RP[i],
                                        &reviewperiod[i], &policy[i], &itemcost[i], &LT[i], &LOC[i], &CD[i], &DC[i],
                                        &d_dailydemand[i], &d_sd_demand[i], &demandtype[i], &d_errorrate[i]);
                                loopControl = err;
                                if (loopControl > 0) {
                                        c[i] = 0.0;
                                        minFound[i] = 0;
                                        minCounted[i] = 0;
                                        td[i].lastDiffIdx = 0;
                                        td[i].min = 1000000000.0;
                                        /*
                                        * Use the newsboy order quantity calculated using the estimated daily demand.
                                        * Four months of stock is set as the maximum stock to hold.  The markup from
                                        * the part cost to the part sale is set at 0.12 and the salvage at 0.01.
                                        */
                                        d_orderlimit[i] = 120.0*((d_sd_demand[i]*0.432432)+d_dailydemand[i]);
                                        if (controlPolicy > 0)
                                                policy[i] = controlPolicy;
                                }
                        }
                        SKUs = i-1;
                }

                err = fopen_s(&fpeab1,"INVENTORY.TXT","w");
                for (i=0;i<SKUs;i++){
                  fprintf(fpeab1,"%d %s %d %.0f %.0f %d %d %.2f %d %d %d %d %.3f %.3f %d %.2f\n",itemid[i], SKU[i],
level[i],
                        d_SCL[i], d_RP[i], reviewperiod[i],
                        policy[i], itemcost[i], LT[i], LOC[i], CD[i], DC[i], d_dailydemand[i],
d_sd_demand[i],demandtype[i],d_errorrate[i]);
                }
                fflush(fpeab1);
                fclose(fpeab1);

                err = fopen_s(&fpeab1,"INVENTORY1.TXT","w");
                for (i=0;i<SKUs;i++){
                  fprintf(fpeab1,"%d %s %d %.0f %.0f %d %d %.2f %d %d %d %d %.3f %.3f %d %.2f\n",itemid[i], SKU[i],
level[i],
                        d_SCL[i], d_RP[i], reviewperiod[i],
                        policy[i], itemcost[i], LT[i], LOC[i], CD[i], DC[i], d_dailydemand[i],
d_sd_demand[i],demandtype[i],d_errorrate[i]);
                }
                fflush(fpeab1);
                fclose(fpeab1);
```

```
		err = fopen_s(&fpeab1,"INVENTORY2.TXT","w");
		for (i=0;i<SKUs;i++){
		  fprintf(fpeab1,"%d %s %d %.0f %.0f %d %d %.2f %d %d %d %d %.3f %.3f %d %.2f\n",itemid[i], SKU[i],
level[i],
				d_SCL[i], d_RP[i], reviewperiod[i],
				policy[i], itemcost[i], LT[i], LOC[i], CD[i], DC[i], d_dailydemand[i],
d_sd_demand[i],demandtype[i],d_errorrate[i]);
		}
		fflush(fpeab1);
		fclose(fpeab1);

/*
Determine the parameter c
*/

		for (i=0;i<SKUs;i++) {
		  err = strcpy_s(parmfile, _countof(parmfile),SKU[i]);
		  err = strcat_s(parmfile, _countof(parmfile),".PRM");
		  err = fopen_s(&fpeab1,parmfile,"w");
		  if (err != 0) {
				printf("Can not open parm file for writing");
		  } else {
			SCL[i] = (int) (d_SCL[i]);
			if (SCL[i] < 0) {
				SCL[i] = 0;
				d_SCL[i] = 0.0;
			}
			RP[i] = (int) (d_RP[i]);
			if (RP[i] < 0) {
				RP[i] = 0;
				d_RP[i] = 0.0;
			}
			d_reviewPeriod[i] = (double) reviewperiod[i];
			if (reviewperiod[i] < 1) {
				reviewperiod[i] = 1;
				d_reviewPeriod[i] = 1.0;
			}
			fprintf(fpeab1,"%d %d %f %f %f %f %f %d %d %d\n",SCL[i], RP[i], -1.0, -1.0, -1.0, 0.0, 0.0,
reviewperiod[i], policy[i], parameters);
		  }
		  fflush(fpeab1);
		  fclose(fpeab1);
		}

		system("C:/awesim/bin/execute.exe INIT");
		for (j=0;j<SKUs;j++) {
			c[j] = readCostNoise(SKU[j],36, 0, &meanCost);
			if (c[j] > 180) // Set the largest perturbation step to 180 (one half the number of days in a year)
				c[j] = 180;
		}
/*
 Parameter c estimated
*/

		for (i=0;i<SKUs;i++) {
			Ck[i] = c[i]/pow(1.0,gamma);
			err = strcpy_s(parmfile, _countof(parmfile),SKU[i]);
			err = strcat_s(parmfile, _countof(parmfile),".PRM");
			err = fopen_s(&fpeab1,parmfile,"w");
			if (err != 0) {
				printf("Can not open parm file for writing");
			} else {
			  SCL[i] = (int) (d_SCL[i]);
			  if (SCL[i] < 0) {
				SCL[i] = 0;
				d_SCL[i] = 0.0;
			  }
			  RP[i] = (int) (d_RP[i]);
			  if (RP[i] < 0) {
```

```
                                RP[i] = 0;
                                d_RP[i] = 0.0;
                        }
                        d_reviewPeriod[i] = (double) reviewperiod[i];
                        if (reviewperiod[i] < 1) {
                                reviewperiod[i] = 1;
                                d_reviewPeriod[i] = 1.0;
                        }
                        fprintf(fpeab1,"%d %d %f %f %f %f %f %d %d %d\n",SCL[i], RP[i], deltas[0], deltas[1], deltas[2],
Ck[i], 0.0, reviewperiod[i], policy[i], parameters);
                }
                fflush(fpeab1);
                fclose(fpeab1);
        }

//      err = strcat_s(command, _countof(command),scenario);
//      system(command);
        system("C:/awesim/bin/execute.exe PLUS");
        system("C:/awesim/bin/execute.exe MINUS");


        for (i=1;i<runs;i++){
                for (j=0;j<SKUs;j++){
                        switch (policy[j]) {
                                case 1:
                                        parameters = 3;
                                        break;
                                case 2:
                                        parameters = 2;
                                        break;
                                case 3:
                                        parameters = 3;
                                        break;
                                case 4:
                                        parameters = 3;
                                        break;
                                case 5:
                                        parameters = 3;
                                        break;
                                case 6:
                                        parameters = 2;
                                        break;
                                case 7:
                                        parameters = 3;
                                        break;
                                case 8:
                                        parameters = 1;
                                        break;
                        }

                        for (k=0;k<parameters;k++) {
                                if (((double) rand()/(double)RAND_MAX)> 0.5)
                                        deltas[k] = (double) 1.0;
                                else
                                        deltas[k] = (double) -1.0;
                        }

                        pluscost[j] = readCost(SKU[j],1);
                        minuscost[j] = readCost(SKU[j],0);
                        Ck[j] = c[j]/pow((i+1),gamma);

                        gradient = 0;
                        for (k=0;k<parameters;k++) {
                                ghats[k]=(pluscost[j]-minuscost[j])/(2*Ck[j]*deltas[k]);
                                gradient = gradient + ghats[k];
                                /* If this is the first iteration, estimate the 'a' parameter.
                                */
                                if (i == 1) {
                                        switch (k) {
                                        case 0:
```

```
//                                                      if (ceil(0.1*d_SCL[j]) > 1)
//                                                              a[j][k] =
((2.0*Ck[j]*ceil(0.2*d_SCL[j]))*(pow((double)(i+A),alpha)))/fabs(pluscost[j]-minuscost[j]+0.01);
//                                                      else
                                                                a[j][k] =
((2.0*Ck[j])*(pow((double)(i+A),alpha)))/fabs(pluscost[j]-minuscost[j]+0.01);
                                                        break;
                                                case 1:
//                                                      if (ceil(0.1*d_RP[j]) > 1)
//                                                              a[j][k] =
((2.0*Ck[j]*ceil(0.2*d_RP[j]))*(pow((double)(i+A),alpha)))/fabs(pluscost[j]-minuscost[j]+0.01);
//                                                      else
                                                                a[j][k] =
((2.0*Ck[j])*(pow((double)(i+A),alpha)))/fabs(pluscost[j]-minuscost[j]+0.01);
                                                        break;
                                                case 2:
//                                                      if (ceil(0.1*d_reviewPeriod[j]) > 1)
//                                                              a[j][k] =
((2.0*Ck[j]*ceil(0.2*d_reviewPeriod[j]))*(pow((double)(i+A),alpha)))/fabs(pluscost[j]-minuscost[j]+0.01);
//                                                      else
                                                                a[j][k] =
((2.0*Ck[j])*(pow((double)(i+A),alpha)))/fabs(pluscost[j]-minuscost[j]+0.01);
                                                        break;
                                                }
                                        }
                                        Ak[j][k] = a[j][k]/pow((double)(i+A),alpha);
                                }

                                if (td[j].min > (pluscost[j]+minuscost[j])/2 ) {
                                        td[j].min = (pluscost[j]+minuscost[j])/2 ;
                                        minFound[j] = 0;
                                        switch (parameters) {
                                        case 1: //(S-1,S,1) policy
                                                td[j].minArgs[0] = d_SCL[j];
                                                td[j].minArgs[1] = d_SCL[j]-1;
                                                td[j].minArgs[2] = (double) 1.0;
                                                break;
                                        case 2:
                                                td[j].minArgs[0] = d_SCL[j];
                                                td[j].minArgs[1] = d_RP[j];
                                                td[j].minArgs[2] = (double) reviewperiod[j];
                                                break;
                                        case 3:
                                                td[j].minArgs[0] = d_SCL[j];
                                                td[j].minArgs[1] = d_RP[j];
                                           td[j].minArgs[2] = d_reviewPeriod[j];
                                                break;
                                        }
                                }

                                if (i > 20 && fabs(((pluscost[j]+minuscost[j])/2) - td[j].min) <= (0.1*itemcost[j])) {
                                        td[j].minArgCount = 0;
                                        if (abs(td[j].minArgs[0]-d_SCL[j]) < .3)
                                                td[j].minArgCount++;
                                        if (abs(td[j].minArgs[1]-d_RP[j]) < .3)
                                                td[j].minArgCount++;
                                        if (abs(td[j].minArgs[2]-d_reviewPeriod[j]) < .3)
                                                td[j].minArgCount++;

                                        if (td[j].minArgCount == 3)
                                                minFound[j]++;
                                        else
                                                if (minFound[j] > 0)
                                                        minFound[j]--;
                                }

                                if (td[j].lastDiffIdx == 10) {
                                        td[j].avgDiff = 0.0;
                                        td[j].madDiff = 0.0;
                                        td[j].movingAvg = 0.0;
```

```
                        for (m=0;m<9;m++) {
                                td[j].costDiff[m] = td[j].costDiff[m+1];
                                td[j].costAvgs[m] = td[j].costAvgs[m+1];
                                td[j].avgDiff = td[j].avgDiff + td[j].costDiff[m];
                                td[j].movingAvg = td[j].movingAvg + td[j].costAvgs[m];
                        }
                        td[j].costDiff[9] = (pluscost[j]-minuscost[j]);
                        td[j].costAvgs[9] = (pluscost[j]+minuscost[j])/2;
                        td[j].movingAvg = td[j].movingAvg + td[j].costAvgs[9];
                        td[j].avgDiff = td[j].avgDiff + td[j].costDiff[9];
                        td[j].movingAvg = td[j].movingAvg/10;
                        td[j].avgDiff = td[j].avgDiff/10;
                        for (m=0;m<9;m++) {
                                td[j].madDiff = td[j].madDiff + fabs(td[j].costDiff[m]-
td[j].costDiff[m+1]);
                        }
                        td[j].madDiff = td[j].madDiff/9;
                }
                else {
                        td[j].movingAvg = 0.0;
                        td[j].costAvgs[td[j].lastDiffIdx] = (pluscost[j]+minuscost[j])/2;
                        td[j].movingAvg = 0.2*td[j].movingAvg+0.8*(pluscost[j]+minuscost[j])/2;
                        td[j].costDiff[td[j].lastDiffIdx++] = (pluscost[j]-minuscost[j]);
                }

                if (gradient <= 0.0001 && i > 10) {
                        if (fabs(((pluscost[j]+minuscost[j])/2) - td[j].min) <= (0.1*itemcost[j])) {
                                minFound[j] = 11;
                        }
                }

                printf("%.2f %5.2f %5.2f %5.2f %5.2f  ",d_SCL[j], d_RP[j], d_reviewPeriod[j], pluscost[j],
minuscost[j]);

                if (minCounted[j] || minFound[j] > 10) {
                        switch (parameters) {
                        case 1: //(S-1,S,1) policy
                                td[j].minArgs[0] = d_SCL[j];
                                if (d_SCL[j] == 0)
                                        td[j].minArgs[1] = d_SCL[j];
                                else
                                        td[j].minArgs[1] = d_SCL[j]-1;
                                td[j].minArgs[2] = (double) 1.0;
                                break;
                        case 2:
                                td[j].minArgs[0] = d_SCL[j];
                                td[j].minArgs[1] = d_RP[j];
                                td[j].minArgs[2] = (double) reviewperiod[j];
                                break;
                        case 3:
                                td[j].minArgs[0] = d_SCL[j];
                                td[j].minArgs[1] = d_RP[j];
                           td[j].minArgs[2] = d_reviewPeriod[j];
                                break;
                        }
                }
                /* This is to help prevent getting stuck in a local minima after a better minimum has already
been seen.
                */
                else if (i > 20 &&
                           (   td[j].min*1.1 < (pluscost[j]+minuscost[j])/2 &&
                                 td[j].madDiff <= ((pluscost[j]+minuscost[j])/2)*0.2 &&
                                 td[j].min*1.1 < td[j].movingAvg )){
                        switch (parameters) {
                        case 1: //(S-1,S,1) policy
                                d_SCL[j] = td[j].minArgs[0];
                                if (d_SCL[j] == 0)
                                        d_RP[j] = d_SCL[j];
                                else
                                        d_RP[j] = d_SCL[j]-1;
```

```
                        d_reviewPeriod[j] = (double) 1.0;
                        break;
                case 2:
                        d_SCL[j] = td[j].minArgs[0];
                        d_RP[j] = td[j].minArgs[1];
                        d_reviewPeriod[j] = (double) reviewperiod[j];
                        break;
                case 3:
                        d_SCL[j] = td[j].minArgs[0];
                        d_RP[j] = td[j].minArgs[1];
                   d_reviewPeriod[j] = td[j].minArgs[2];
                        break;
                }
        }
        /* This is to prevent selection of a parameter set that yeilds a much higher (worse) cost funtion
result.
        */
        else if (i>20 && td[j].min*2 < (pluscost[j]+minuscost[j])/2) {
                switch (parameters) {
                case 1: //(S-1,S,1) policy
                        d_SCL[j] = td[j].minArgs[0];
                        if (d_SCL[j] == 0)
                                d_RP[j] = d_SCL[j];
                        else
                                d_RP[j] = d_SCL[j]-1;
                        d_RP[j] = d_SCL[j]-1;
                        d_reviewPeriod[j] = (double) 1.0;
                        break;
                case 2:
                        d_SCL[j] = td[j].minArgs[0];
                        d_RP[j] = td[j].minArgs[1];
                        d_reviewPeriod[j] = (double) reviewperiod[j];
                        break;
                case 3:
                        d_SCL[j] = td[j].minArgs[0];
                        d_RP[j] = td[j].minArgs[1];
                   d_reviewPeriod[j] = td[j].minArgs[2];
                        break;
                }
        }
        else {
                switch (parameters) {
                case 1:
                        d_SCL[j] = (d_SCL[j]-Ak[j][0]*ghats[0]);
                        break;
                case 2:
                        d_SCL[j] = (d_SCL[j]-Ak[j][0]*ghats[0]);
                        d_RP[j] = (d_RP[j]-Ak[j][1]*ghats[1]);
                        break;
                case 3:
                        d_SCL[j] = (d_SCL[j]-Ak[j][0]*ghats[0]);
                        d_RP[j] = (d_RP[j]-Ak[j][1]*ghats[1]);
                        d_reviewPeriod[j] = (d_reviewPeriod[j]-Ak[j][2]*ghats[2]);
                        break;
                }
        }

        if (d_SCL[j] > d_orderlimit[j])
                d_SCL[j] = d_orderlimit[j];

        if (d_RP[j] > d_orderlimit[j])
                d_RP[j] = d_orderlimit[j];

        err = strcpy_s(parmfile, _countof(parmfile),SKU[j]);
        err = strcat_s(parmfile, _countof(parmfile),".PRM");
        err = fopen_s(&fpeab1,parmfile,"w");
        if (err != 0) {
                 printf("Can not open parm file for writing");
        } else {
                switch (parameters) {
```

```
		case 1:
			if (d_SCL[j]-floor(d_SCL[j])> 0.5 )
				SCL[j] = (int) ceil(d_SCL[j]);
			else
				SCL[j] = (int) floor(d_SCL[j]);
			// Base stock policy, order same day as demand
			if (d_SCL[j] == 0)
				d_RP[j] = d_SCL[j];
			else
				d_RP[j] = d_SCL[j]-1;
			d_reviewPeriod[j] = 1.0;
			if (SCL[j] == 0)
				RP[j] = SCL[j];
			else
				RP[j] = SCL[j]-1;
			reviewperiod[j] = 1;
			break;
		case 2:
			if (d_SCL[j]-floor(d_SCL[j])> 0.5 )
				SCL[j] = (int) ceil(d_SCL[j]);
			else
				SCL[j] = (int) floor(d_SCL[j]);
			if (d_RP[j]-floor(d_RP[j])>0.5)
				RP[j] = (int) ceil(d_RP[j]);
			else
				RP[j] = (int) floor(d_RP[j]);
			break;
		case 3:
			if (d_SCL[j]-floor(d_SCL[j])> 0.5 )
				SCL[j] = (int) ceil(d_SCL[j]);
			else
				SCL[j] = (int) floor(d_SCL[j]);
			if (d_RP[j]-floor(d_RP[j])>0.5)
				RP[j] = (int) ceil(d_RP[j]);
			else
				RP[j] = (int) floor(d_RP[j]);
			if (d_reviewPeriod[j]-floor(d_reviewPeriod[j])>0.5)
				reviewperiod[j] = (int) ceil(d_reviewPeriod[j]);
			else
				reviewperiod[j] = (int) floor(d_reviewPeriod[j]);

			switch (policy[j]){
			case 3: // rQ review period policy
				RP[j] = SCL[j];
				d_RP[j] = d_SCL[j];
				break;
			case 4: // Modified 2 bin policy
				RP[j] = (int) ceil((double) (SCL[j])*0.5);
				d_RP[j] = d_SCL[j]*0.5;
				break;
			case 7: // sS review period policy
				RP[j] = SCL[j];
				d_RP[j] = d_SCL[j];
				break;
			}
			break;
		}
		if (policy[j] > 4 && d_RP[j]>d_SCL[j]) // These are the Ss policies
			d_RP[j] = d_SCL[j];

	if (SCL[j] < 0) {
		SCL[j] = 0;
		d_SCL[j] = 0.0;
	}
	if (RP[j] < 0 || SCL[j] == 0) {
		RP[j] = 0;
		d_RP[j] = 0.0;
	}
	if (reviewperiod[j] < 1) {
		reviewperiod[j] = 1;
```

```
                                        d_reviewPeriod[j] = 1.0;
                                }
                                if (reviewperiod[j] > 365) {
                                        reviewperiod[j] = 360;
                                        d_reviewPeriod[j] = 360.0;
                                }

                                if (minFound[j] > 10 && minCounted[j] == 0) {
                                        minCounted[j] = 1;
                                        minsFound++;
                                }

                                        fprintf(fpeab1,"%d %d %f %f %f %f %f %d %d %d\n",SCL[j], RP[j], deltas[0],
deltas[1], deltas[2], Ck[j], Ak[j][0], reviewperiod[j], policy[j], parameters);
                                }
                                fflush(fpeab1);
                                fclose(fpeab1);
                        }
                        printf("\n");
                        if (minsFound < SKUs){
                                system("C:/awesim/bin/execute.exe PLUS");
                                system("C:/awesim/bin/execute.exe MINUS");
                        }
                        else
                                break;
                }
                printf("\n");
                printf("Mins Found %d",minsFound);

                err = fopen_s(&fpeab,"INVENTORYOPT.TXT","w");
                if (err != 0) {
                        printf("Can not open SKU file for writing");
                } else {
                        for (i=0;i<SKUs;i++){
                          fprintf(fpeab,"%d %s %d %.0f %.0f %.0f %d %.2f %d %d %d %d %.3f %.3f %d %.2f\n",itemid[i],
SKU[i], level[i],
                          td[i].minArgs[0], td[i].minArgs[1], td[i].minArgs[2],
                          policy[i], itemcost[i], LT[i], LOC[i], CD[i], DC[i], d_dailydemand[i], d_sd_demand[i], demandtype[i],
d_errorrate[i]);
                        }
                        fflush(fpeab);
                        fclose(fpeab);
                }

                system("C:/awesim/bin/execute.exe OPT");

                err = strcat_s(optout, _countof(optout),inventory);
                err = fopen_s(&fpeab1,optout,"w");
                if (err != 0) {
                        printf("Can not open Optimal cost file for writing");
                } else {
                        for (i=0;i<SKUs;i++){
                                c[i] = readCostNoise(SKU[i], 36, 1, &meanCost);
                                fprintf(fpeab1, "%s, %d, %.2lf, %.2lf, %.3lf, %d, %.2f, %.0f, %.0f, %.0f\n",SKU[i], policy[i],
                                        itemcost[i], meanCost, c[i], demandtype[i], d_errorrate[i], td[i].minArgs[0],
td[i].minArgs[1], td[i].minArgs[2]);
                        }
                        fflush(fpeab1);
                        fclose(fpeab1);
                }


                return 0;
}
```

Appendix E

ANOVA Tables and diagnostics for the SPSA simulation optimization experiments

Aircraft data inventory cost mean

Response: Cost Transform: Natural log Constant: 0
ANOVA for Selected Factorial Model
Analysis of variance table [Partial sum of squares]

Source	Sum of Squares	DF	Mean Square	F Value	Prob > F	
Model	1372.96	63	21.79	3.12	< 0.0001	significant
Review A	*0.44*	*1*	*0.44*	*0.063*	*0.8020*	
Demand B	*315.70*	*3*	*105.23*	*15.06*	*< 0.0001*	significant
Policy C	*399.28*	*7*	*57.04*	*8.16*	*< 0.0001*	significant
AB	*60.49*	*3*	*20.16*	*2.88*	*0.0345*	significant
AC	*129.20*	*7*	*18.46*	*2.64*	*0.0101*	significant
BC	*288.82*	*21*	*13.75*	*1.97*	*0.0054*	significant
ABC	*179.03*	*21*	*8.53*	*1.22*	*0.2228*	
Pure Error	19682.82	2816	6.99			
Cor Total	21055.78	2879				

The Model F-value of 3.12 implies the model is significant. There is only a 0.01% chance that a "Model F-Value" this large could occur due to noise.

Values of "Prob > F" less than 0.0500 indicate model terms are significant.
In this case B, C, AB, AC, BC are significant model terms.
Values greater than 0.1000 indicate the model terms are not significant.
If there are many insignificant model terms (not counting those required to support hierarchy), model reduction may improve your model.

Std. Dev.	2.64	R-Squared	0.0652
Mean	3.41	Adj R-Squared	0.0443
C.V.	77.54	Pred R-Squared	0.0222
PRESS	20587.66	Adeq Precision	8.385

The "Pred R-Squared" of 0.0222 is in reasonable agreement with the "Adj R-Squared" of 0.0443.

"Adeq Precision" measures the signal to noise ratio. A ratio greater than 4 is desirable. Your ratio of 8.385 indicates an adequate signal. This model can be used to navigate the design space.

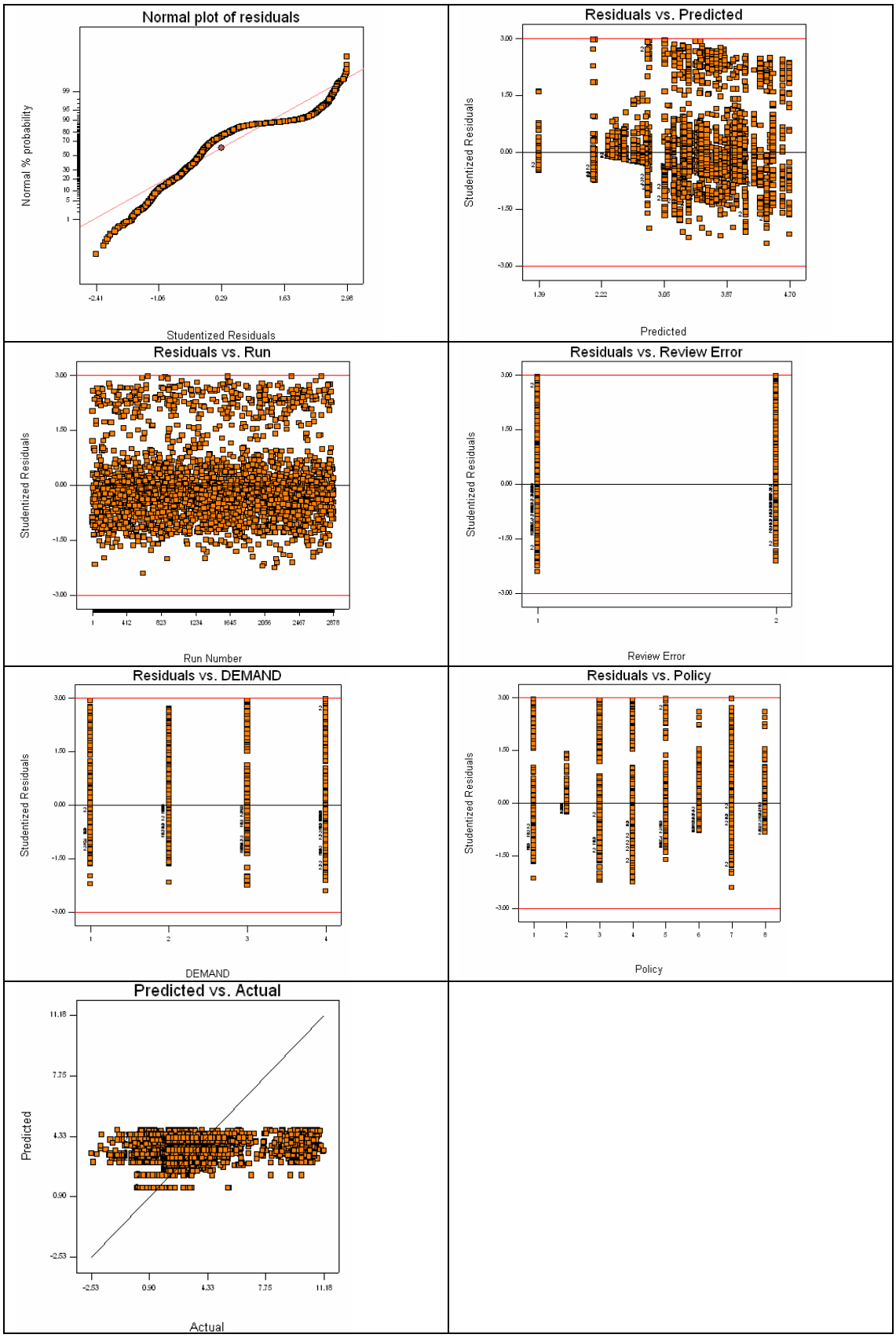


Figure 27: Aircraft data inventory cost mean ANOVA diagnostics

Aircraft data inventory optimization delta

Response: OptDelta
ANOVA for Selected Factorial Model
Analysis of variance table [Partial sum of squares]

Source	Sum of Squares	DF	Mean Square	F Value	Prob > F	
Model	1.809E+010	63	2.872E+008	3.76	< 0.0001	significant
Review A	*7.848E+007*	*1*	*7.848E+007*	*1.03*	*0.3110*	
Demand B	*1.855E+008*	*3*	*6.185E+007*	*0.81*	*0.4887*	
Policy C	*1.356E+010*	*7*	*1.937E+009*	*25.35*	*< 0.0001*	significant
AB	*3.107E+008*	*3*	*1.036E+008*	*1.35*	*0.2548*	
AC	*9.019E+008*	*7*	*1.288E+008*	*1.69*	*0.1079*	
BC	*2.135E+009*	*21*	*1.017E+008*	*1.33*	*0.1434*	
ABC	*9.202E+008*	*21*	*4.382E+007*	*0.57*	*0.9382*	
Pure Error	2.152E+011	2816	7.644E+007			
Cor Total	2.333E+011	2879				

The Model F-value of 3.76 implies the model is significant. There is only a 0.01% chance that a "Model F-Value" this large could occur due to noise.

Values of "Prob > F" less than 0.0500 indicate model terms are significant.
In this case C are significant model terms.
Values greater than 0.1000 indicate the model terms are not significant.
If there are many insignificant model terms (not counting those required to support hierarchy), model reduction may improve your model.

Std. Dev.	8742.83	R-Squared	0.0775
Mean1	858.59	Adj R-Squared	0.0569
C.V.470.40		Pred R-Squared	0.0351
PRESS	2.251E+011	Adeq Precision	6.491

The "Pred R-Squared" of 0.0351 is in reasonable agreement with the "Adj R-Squared" of 0.0569.

"Adeq Precision" measures the signal to noise ratio. A ratio greater than 4 is desirable. Your ratio of 6.491 indicates an adequate signal. This model can be used to navigate the design space.

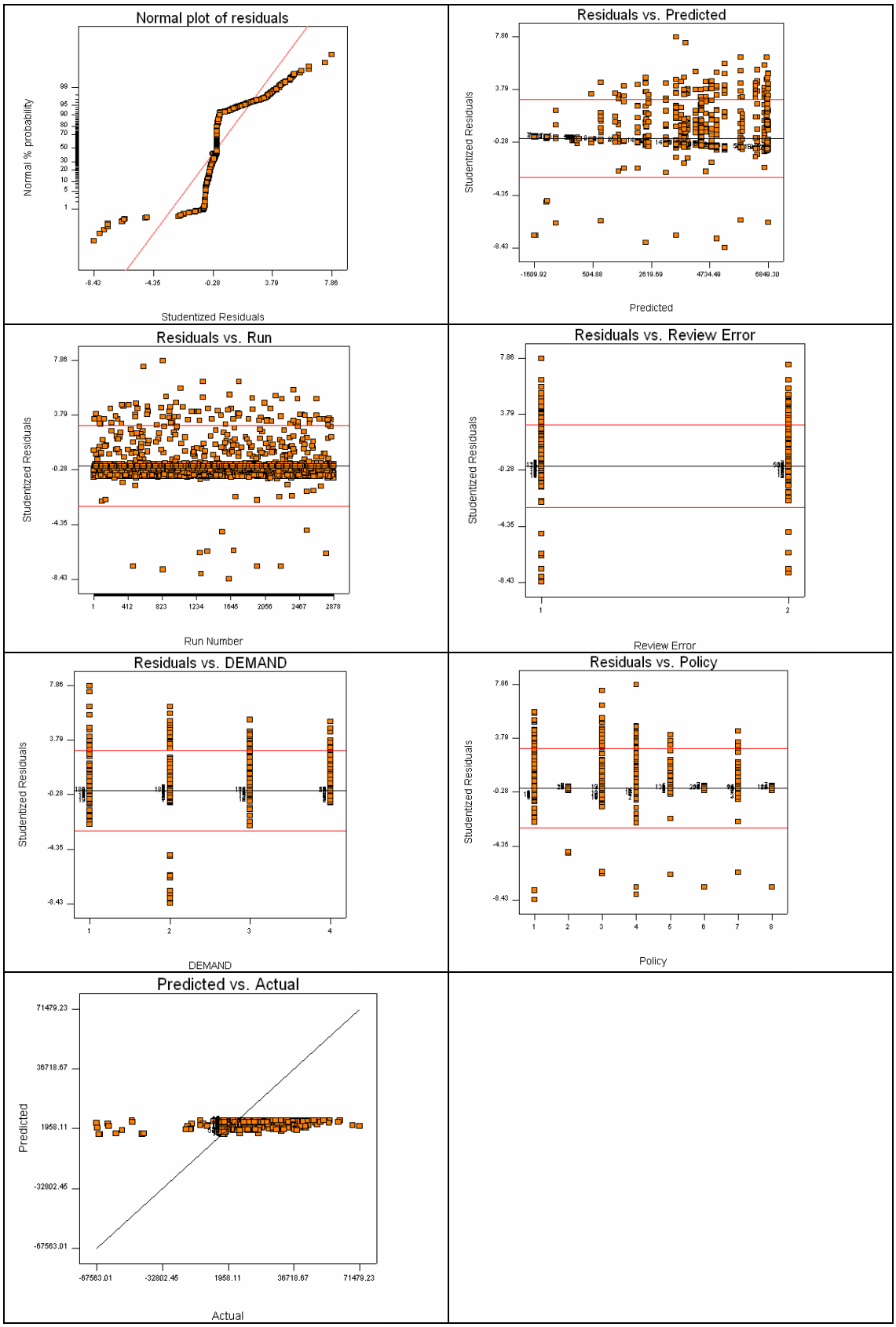


Figure 28: Aircraft data optimization delta ANOVA diagnostics

Aircraft data inventory cost standard deviation

Response: StdDev Transform: Power Lambda: -2.75 Constant: 453.889

ANOVA for Selected Factorial Model

Analysis of variance table [Partial sum of squares]

Source	Sum of Squares	DF	Mean Square	F Value	Prob > F	
Model	1.109E-013	63	1.761E-015	8.32	< 0.0001	significant
A	*2.969E-016*	*1*	*2.969E-016*	*1.40*	*0.2363*	
B	*1.166E-015*	*3*	*3.885E-016*	*1.84*	*0.1385*	
C	*8.664E-014*	*7*	*1.238E-014*	*58.48*	*< 0.0001*	
AB	*2.976E-015*	*3*	*9.921E-016*	*4.69*	*0.0029*	
AC	*2.128E-015*	*7*	*3.040E-016*	*1.44*	*0.1860*	
BC	*9.669E-015*	*21*	*4.604E-016*	*2.18*	*0.0015*	
ABC	*8.064E-015*	*21*	*3.840E-016*	*1.81*	*0.0130*	
Pure Error	5.960E-013	2816	2.116E-016			
Cor Total	7.069E-013	2879				

The Model F-value of 8.32 implies the model is significant. There is only a 0.01% chance that a "Model F-Value" this large could occur due to noise.

Values of "Prob > F" less than 0.0500 indicate model terms are significant.
In this case C, AB, BC, ABC are significant model terms.
Values greater than 0.1000 indicate the model terms are not significant.
If there are many insignificant model terms (not counting those required to support hierarchy), model reduction may improve your model.

Std. Dev.	1.455E-008	R-Squared	0.1569
Mean	4.293E-008	Adj R-Squared	0.1381
C.V.	33.89	Pred R-Squared	0.1182
PRESS	6.234E-013	Adeq Precision	8.966

The "Pred R-Squared" of 0.1182 is in reasonable agreement with the "Adj R-Squared" of 0.1381.

"Adeq Precision" measures the signal to noise ratio. A ratio greater than 4 is desirable. Your ratio of 8.966 indicates an adequate signal. This model can be used to navigate the design space.

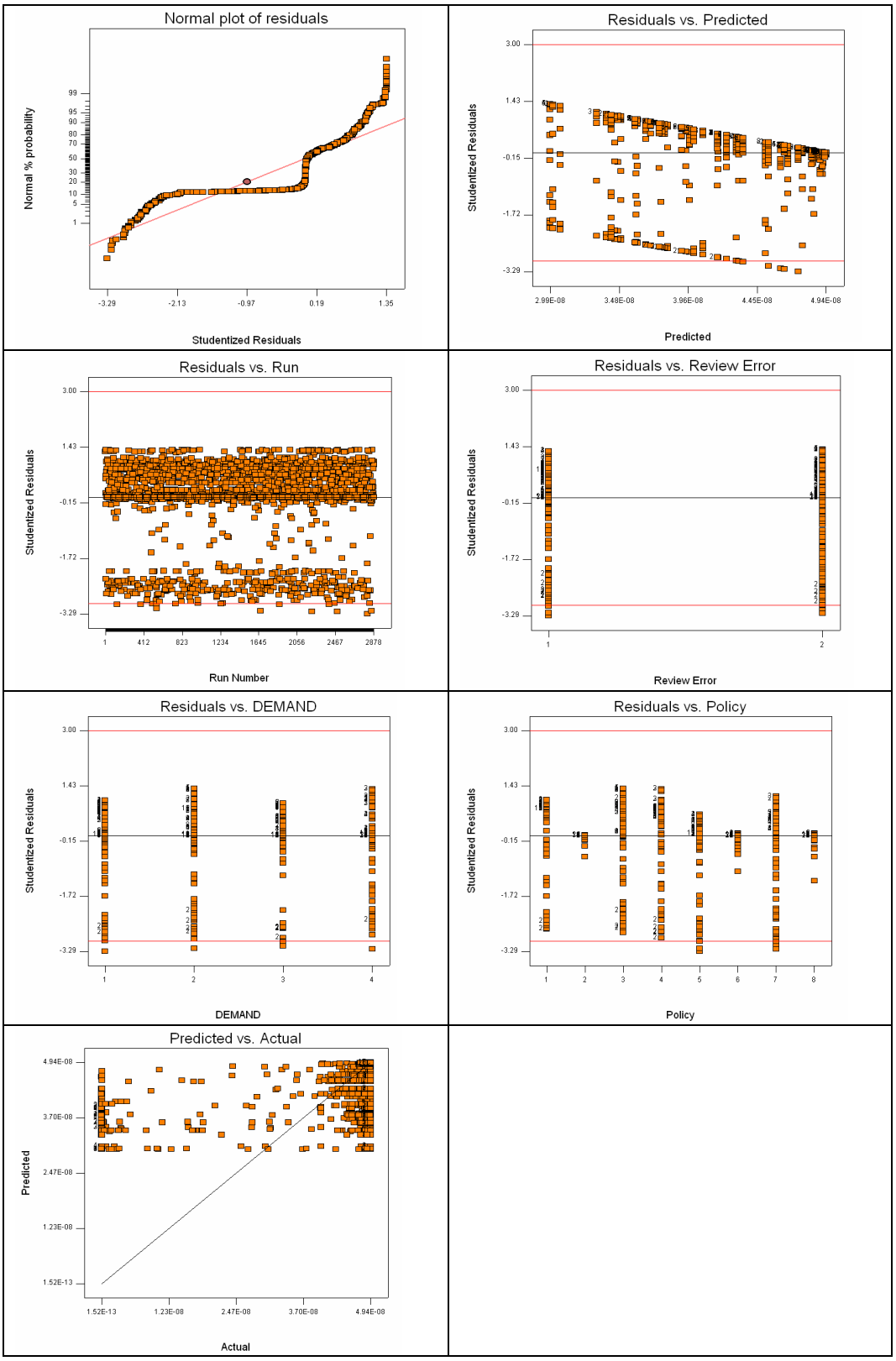


Figure 29: Aircraft data inventory cost standard deviation ANOVA diagnostics

Vehicle data, large workforce inventory cost mean

Response: Cost Transform: Natural log Constant: 0
ANOVA for Selected Factorial Model
Analysis of variance table [Partial sum of squares]

Source	Sum of Squares	DF	Mean Square	F Value	Prob > F	
Model	530.03	63	8.41	1.31	0.0613	not significant
Review A	*4.87*	*1*	*4.87*	*0.76*	*0.3840*	
Demand B	*176.92*	*3*	*58.97*	*9.19*	*< 0.0001*	significant
Policy C	*234.80*	*7*	*33.54*	*5.23*	*< 0.0001*	significant
AB	*3.95*	*3*	*1.32*	*0.21*	*0.8929*	
AC	*16.57*	*7*	*2.37*	*0.37*	*0.9203*	
BC	*58.35*	*21*	*2.78*	*0.43*	*0.9878*	
ABC	*34.57*	*21*	*1.65*	*0.26*	*0.9997*	
Pure Error	3696.64	576	6.42			
Cor Total	4226.67	639				

The Model F-value of 1.31 implies there is a 6.13% chance that a "Model F-Value" this large could occur due to noise.

Values of "Prob > F" less than 0.0500 indicate model terms are significant.
In this case B, C are significant model terms.
Values greater than 0.1000 indicate the model terms are not significant.
If there are many insignificant model terms (not counting those required to support hierarchy), model reduction may improve your model.

Std. Dev.	2.53	R-Squared	0.1254
Mean	8.26	Adj R-Squared	0.0297
C.V.	30.69	Pred R-Squared	-0.0798
PRESS	4563.75	Adeq Precision	5.685

A negative "Pred R-Squared" implies that the overall mean is a better predictor of your response than the current model.

"Adeq Precision" measures the signal to noise ratio. A ratio greater than 4 is desirable. Your ratio of 5.685 indicates an adequate signal. This model can be used to navigate the design space.

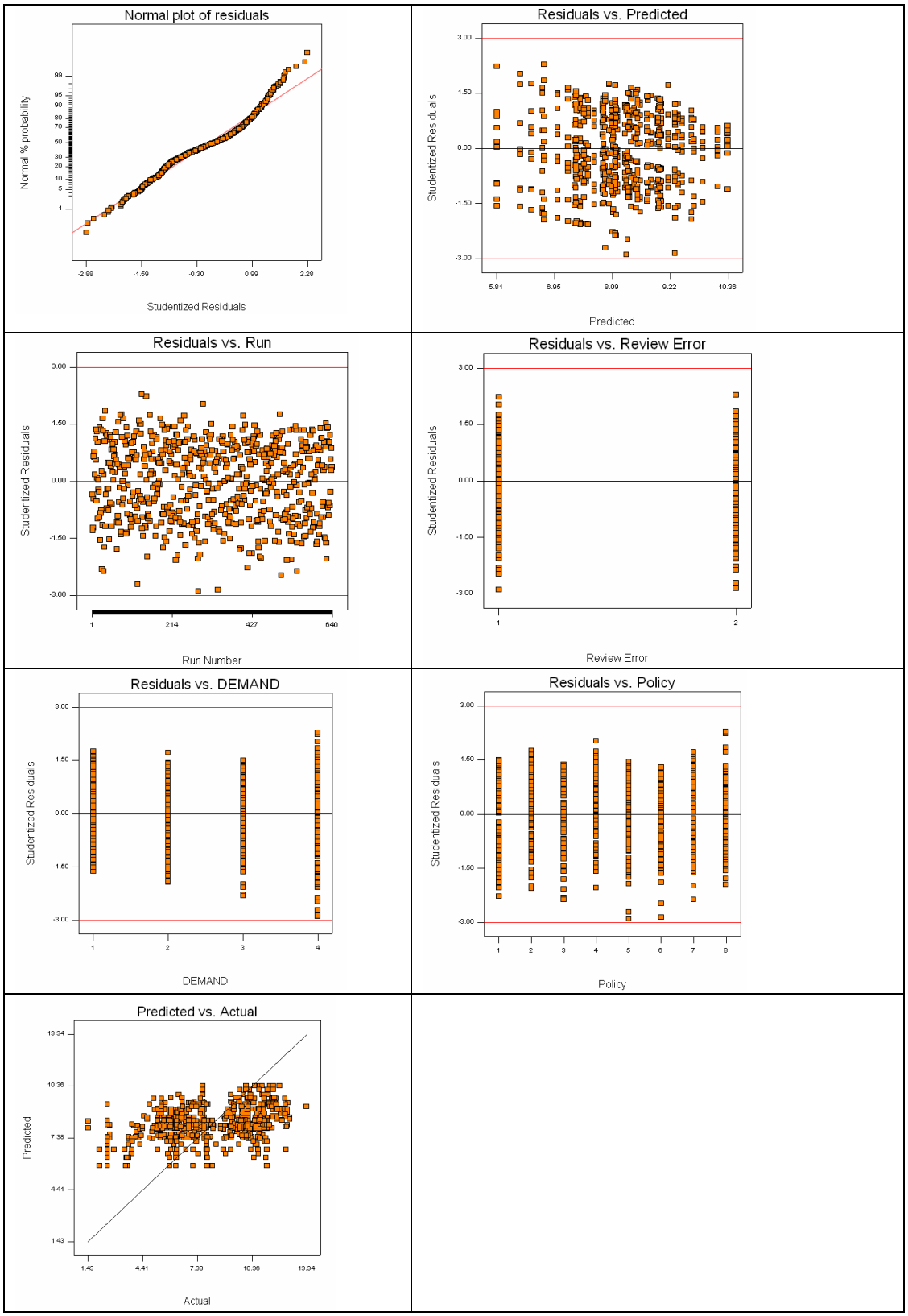


Figure 30: Large workforce vehicle data inventory cost mean ANOVA diagnostics

Vehicle data, large workforce optimization delta

Response: OptDelta
ANOVA for Selected Factorial Model
Analysis of variance table [Partial sum of squares]

Source	Sum of Squares	DF	Mean Square	F Value	Prob > F	
Model	4.516E+012	63	7.167E+010	6.55	< 0.0001	significant
Review A	*8.718E+009*	*1*	*8.718E+009*	*0.80*	*0.3726*	
Demand B	*1.266E+012*	*3*	*4.220E+011*	*38.54*	*< 0.0001*	significant
Policy C	*2.583E+012*	*7*	*3.689E+011*	*33.69*	*< 0.0001*	significant
AB	*5.270E+009*	*3*	*1.757E+009*	*0.16*	*0.9229*	
AC	*1.479E+010*	*7*	*2.113E+009*	*0.19*	*0.9869*	
BC	*6.195E+011*	*21*	*2.950E+010*	*2.69*	*< 0.0001*	significant
ABC	*1.872E+010*	*21*	*8.916E+008*	*0.081*	*1.0000*	
Pure Error	6.307E+012	576	1.095E+010			
Cor Total	1.082E+013	639				

The Model F-value of 6.55 implies the model is significant. There is only a 0.01% chance that a "Model F-Value" this large could occur due to noise.

Values of "Prob > F" less than 0.0500 indicate model terms are significant.
In this case B, C, BC are significant model terms.
Values greater than 0.1000 indicate the model terms are not significant.
If there are many insignificant model terms (not counting those required to support hierarchy), model reduction may improve your model.

Std. Dev.	1.046E+005	R-Squared	0.4172
Mean	-87080.63	Adj R-Squared	0.3535
C.V.	-120.17	Pred R-Squared	0.2805
PRESS	7.786E+012	Adeq Precision	9.161

The "Pred R-Squared" of 0.2805 is in reasonable agreement with the "Adj R-Squared" of 0.3535.

"Adeq Precision" measures the signal to noise ratio. A ratio greater than 4 is desirable. Your ratio of 9.161 indicates an adequate signal. This model can be used to navigate the design space.

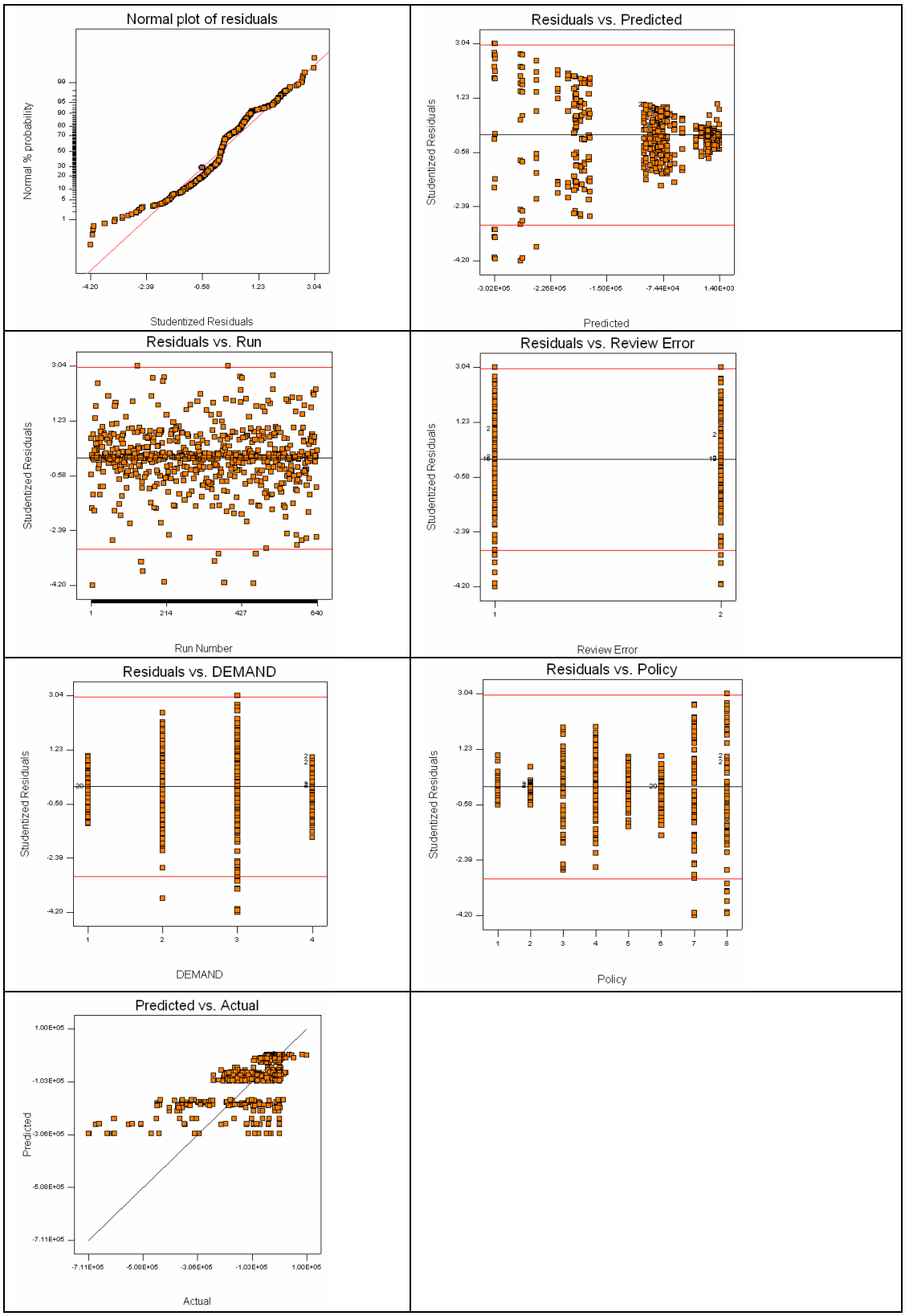


Figure 31: Large workforce vehicle data optimization delta ANOVA diagnostics

Vehicle data, large workforce inventory cost standard deviation

Response: StdDev Transform: Inverse sqrt Constant: 582.638

ANOVA for Selected Factorial Model

Analysis of variance table [Partial sum of squares]

Source	Sum of Squares	DF	Mean Square	F Value	Prob > F	
Model	0.016	42	3.912E-004	2.08	0.0001	significant
A	*5.062E-004*	*1*	*5.062E-004*	*2.69*	*0.1014*	
B	*6.588E-003*	*3*	*2.196E-003*	*11.67*	*< 0.0001*	
C	*4.927E-003*	*7*	*7.039E-004*	*3.74*	*0.0006*	
AB	*3.153E-004*	*3*	*1.051E-004*	*0.56*	*0.6425*	
AC	*3.843E-004*	*7*	*5.490E-005*	*0.29*	*0.9571*	
BC	*3.708E-003*	*21*	*1.766E-004*	*0.94*	*0.5403*	
Residual	0.11	597	1.881E-004			
Lack of Fit	*1.094E-003*	*21*	*5.209E-005*	*0.27*	*0.9996*	*not significant*
Pure Error	*0.11*	*576*	*1.931E-004*			
Cor Total	0.13	639				

The Model F-value of 2.08 implies the model is significant. There is only a 0.01% chance that a "Model F-Value" this large could occur due to noise.

Values of "Prob > F" less than 0.0500 indicate model terms are significant.
In this case B, C are significant model terms.
Values greater than 0.1000 indicate the model terms are not significant.
If there are many insignificant model terms (not counting those required to support hierarchy), model reduction may improve your model.

The "Lack of Fit F-value" of 0.27 implies the Lack of Fit is not significant relative to the pure error. There is a 99.96% chance that a "Lack of Fit F-value" this large could occur due to noise. Non-significant lack of fit is good -- we want the model to fit.

Std. Dev.	0.014	R-Squared	0.1276
Mean	0.030	Adj R-Squared	0.0662
C.V.	45.85	Pred R-Squared	-0.0026
PRESS	0.13	Adeq Precision	6.531

A negative "Pred R-Squared" implies that the overall mean is a better predictor of your response than the current model.

"Adeq Precision" measures the signal to noise ratio. A ratio greater than 4 is desirable. Your ratio of 6.531 indicates an adequate signal. This model can be used to navigate the design space.

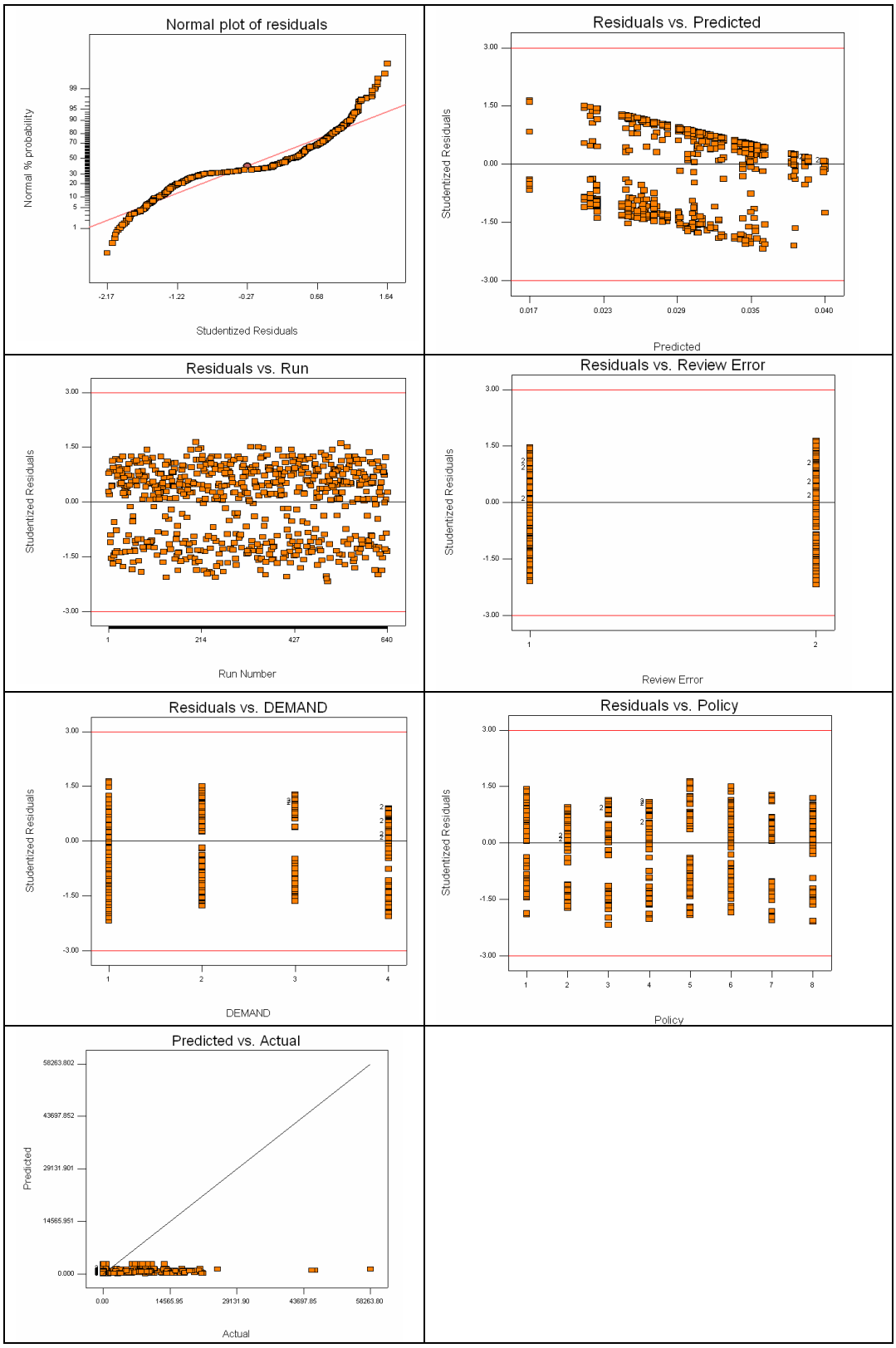


Figure 32: Large workforce vehicle data inventory cost standard deviation ANOVA diagnostics

Vehicle data, small workforce inventory cost mean

Response: Cost Transform: Natural log Constant: 0

ANOVA for Selected Factorial Model

Analysis of variance table [Partial sum of squares]

Source	Sum of Squares	DF	Mean Square	F Value	Prob > F	
Model	476.14	42	11.34	2.63	< 0.0001	significant
A	*1.68*	*1*	*1.68*	*0.39*	*0.5324*	
B	*266.08*	*3*	*88.69*	*20.57*	*< 0.0001*	
C	*132.44*	*7*	*18.92*	*4.39*	*< 0.0001*	
AB	*6.90*	*3*	*2.30*	*0.53*	*0.6596*	
AC	*14.97*	*7*	*2.14*	*0.50*	*0.8379*	
BC	*54.08*	*21*	*2.58*	*0.60*	*0.9218*	
Residual	2574.58	597	4.31			
Lack of Fit	*12.16*	*21*	*0.58*	*0.13*	*1.0000*	*not significant*
Pure Error	*2562.42*	*576*	*4.45*			
Cor Total	3050.72	639				

The Model F-value of 2.63 implies the model is significant. There is only a 0.01% chance that a "Model F-Value" this large could occur due to noise.

Values of "Prob > F" less than 0.0500 indicate model terms are significant.
In this case B, C are significant model terms.
Values greater than 0.1000 indicate the model terms are not significant.
If there are many insignificant model terms (not counting those required to support hierarchy), model reduction may improve your model.

The "Lack of Fit F-value" of 0.13 implies the Lack of Fit is not significant relative to the pure error. There is a 100.00% chance that a "Lack of Fit F-value" this large could occur due to noise. Non-significant lack of fit is good -- we want the model to fit.

Std. Dev.	2.08	R-Squared	0.1561
Mean	7.61	Adj R-Squared	0.0967
C.V.	27.31	Pred R-Squared	0.0301
PRESS	2958.81	Adeq Precision	6.466

The "Pred R-Squared" of 0.0301 is in reasonable agreement with the "Adj R-Squared" of 0.0967.

"Adeq Precision" measures the signal to noise ratio. A ratio greater than 4 is desirable. Your ratio of 6.466 indicates an adequate signal. This model can be used to navigate the design space.

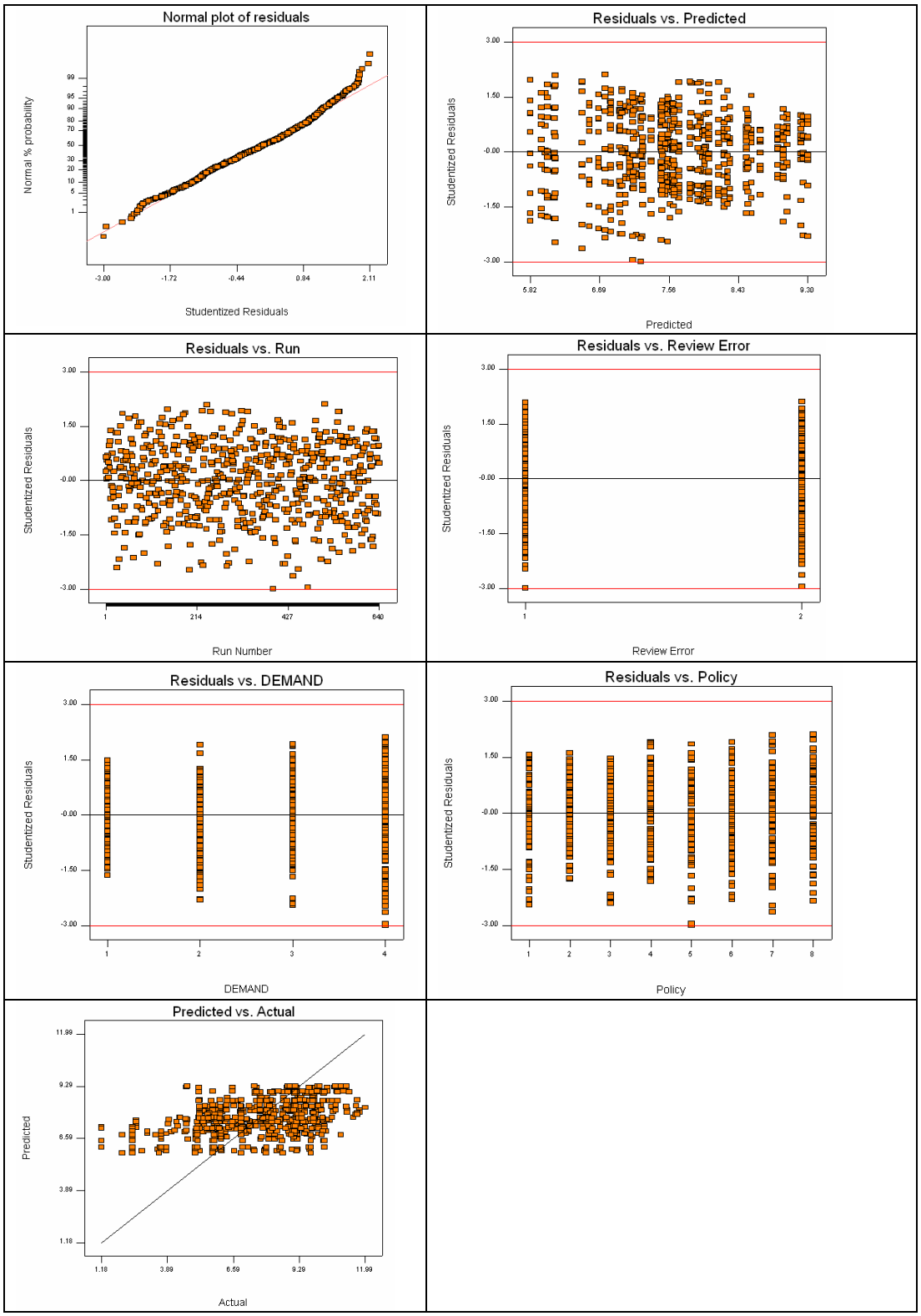


Figure 33: Small workforce vehicle data inventory cost mean ANOVA diagnostics

Vehicle data, small workforce optimization delta

Response: Opt Delta
ANOVA for Selected Factorial Model
Analysis of variance table [Partial sum of squares]

Source	Sum of Squares	DF	Mean Square	F Value	Prob > F	
Model	2.621E+011	42	6.241E+009	7.03	< 0.0001	significant
A	*9.917E+008*	*1*	*9.917E+008*	*1.12*	*0.2910*	
B	*8.349E+010*	*3*	*2.783E+010*	*31.35*	*< 0.0001*	
C	*1.260E+011*	*7*	*1.800E+010*	*20.28*	*< 0.0001*	
AB	*7.689E+008*	*3*	*2.563E+008*	*0.29*	*0.8336*	
AC	*7.564E+008*	*7*	*1.081E+008*	*0.12*	*0.9968*	
BC	*5.012E+010*	*21*	*2.386E+009*	*2.69*	*< 0.0001*	
Residual	5.300E+011	597	8.877E+008			
Lack of Fit	*1.331E+009*	*21*	*6.338E+007*	*0.069*	*1.0000*	*not significant*
Pure Error	*5.286E+011*	*576*	*9.178E+008*			
Cor Total	7.921E+011	639				

The Model F-value of 7.03 implies the model is significant. There is only a 0.01% chance that a "Model F-Value" this large could occur due to noise.

Values of "Prob > F" less than 0.0500 indicate model terms are significant.
In this case B, C, BC are significant model terms.
Values greater than 0.1000 indicate the model terms are not significant.
If there are many insignificant model terms (not counting those required to support hierarchy), model reduction may improve your model.

The "Lack of Fit F-value" of 0.07 implies the Lack of Fit is not significant relative to the pure error. There is a 100.00% chance that a "Lack of Fit F-value" this large could occur due to noise. Non-significant lack of fit is good -- we want the model to fit.

Std. Dev.	29794.46	R-Squared	0.3309
Mean	-18826.33	Adj R-Squared	0.2839
C.V.	-158.26	Pred R-Squared	0.2311
PRESS	6.091E+011	Adeq Precision	9.797

The "Pred R-Squared" of 0.2311 is in reasonable agreement with the "Adj R-Squared" of 0.2839.

"Adeq Precision" measures the signal to noise ratio. A ratio greater than 4 is desirable. Your ratio of 9.797 indicates an adequate signal. This model can be used to navigate the design space.

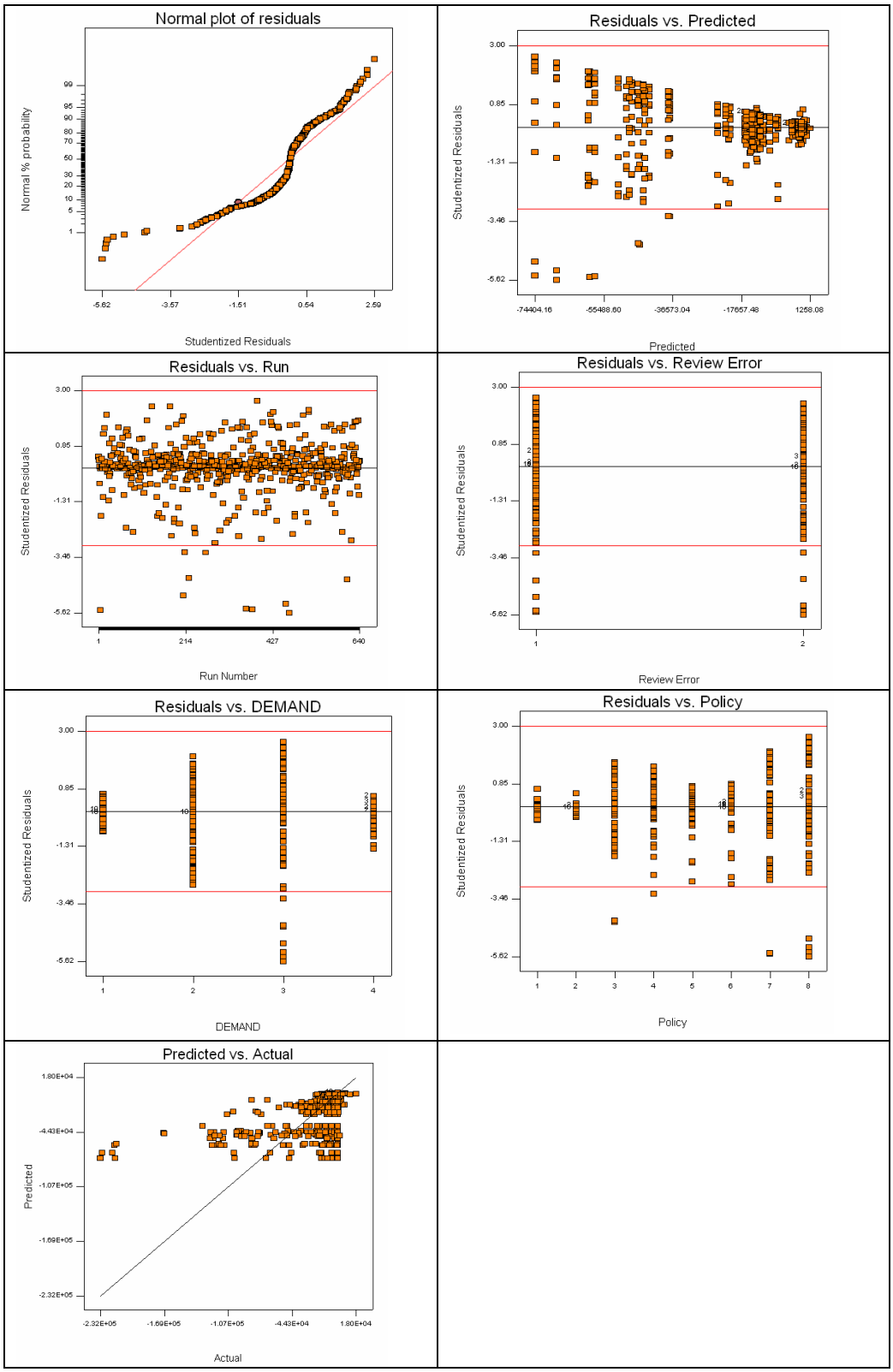


Figure 34: Small workforce vehicle data optimization delta ANOVA diagnostics

Vehicle data, small workforce inventory cost standard deviation

Response: StdDev Transform: Inverse
ANOVA for Selected Factorial Model
Analysis of variance table [Partial sum of squares]

Source	Sum of Squares	DF	Mean Square	F Value	Prob > F	
Model	4.220E-004	42	1.005E-005	3.10	< 0.0001	significant
A	*1.627E-005*	*1*	*1.627E-005*	*5.02*	*0.0255*	
B	*1.563E-004*	*3*	*5.210E-005*	*16.06*	*< 0.0001*	
C	*1.560E-004*	*7*	*2.228E-005*	*6.87*	*< 0.0001*	
AB	*1.732E-005*	*3*	*5.773E-006*	*1.78*	*0.1498*	
AC	*1.244E-005*	*7*	*1.777E-006*	*0.55*	*0.7982*	
BC	*6.366E-005*	*21*	*3.031E-006*	*0.93*	*0.5456*	
Residual	1.936E-003	597	3.243E-006			
Lack of Fit	*1.921E-005*	*21*	*9.149E-007*	*0.27*	*0.9995*	*not significant*
Pure Error	*1.917E-003*	*576*	*3.328E-006*			
Cor Total	2.358E-003	639				

The Model F-value of 3.10 implies the model is significant. There is only a 0.01% chance that a "Model F-Value" this large could occur due to noise.

Values of "Prob > F" less than 0.0500 indicate model terms are significant.
In this case A, B, C are significant model terms.
Values greater than 0.1000 indicate the model terms are not significant.
If there are many insignificant model terms (not counting those required to support hierarchy), model reduction may improve your model.

The "Lack of Fit F-value" of 0.27 implies the Lack of Fit is not significant relative to the pure error. There is a 99.95% chance that a "Lack of Fit F-value" this large could occur due to noise. Non-significant lack of fit is good -- we want the model to fit.

Std. Dev.	1.801E-003	R-Squared	0.1789
Mean	3.108E-003	Adj R-Squared	0.1212
C.V.	57.94	Pred R-Squared	0.0564
PRESS	2.225E-003	Adeq Precision	7.326

The "Pred R-Squared" of 0.0564 is in reasonable agreement with the "Adj R-Squared" of 0.1212.

"Adeq Precision" measures the signal to noise ratio. A ratio greater than 4 is desirable. Your ratio of 7.326 indicates an adequate signal. This model can be used to navigate the design space.

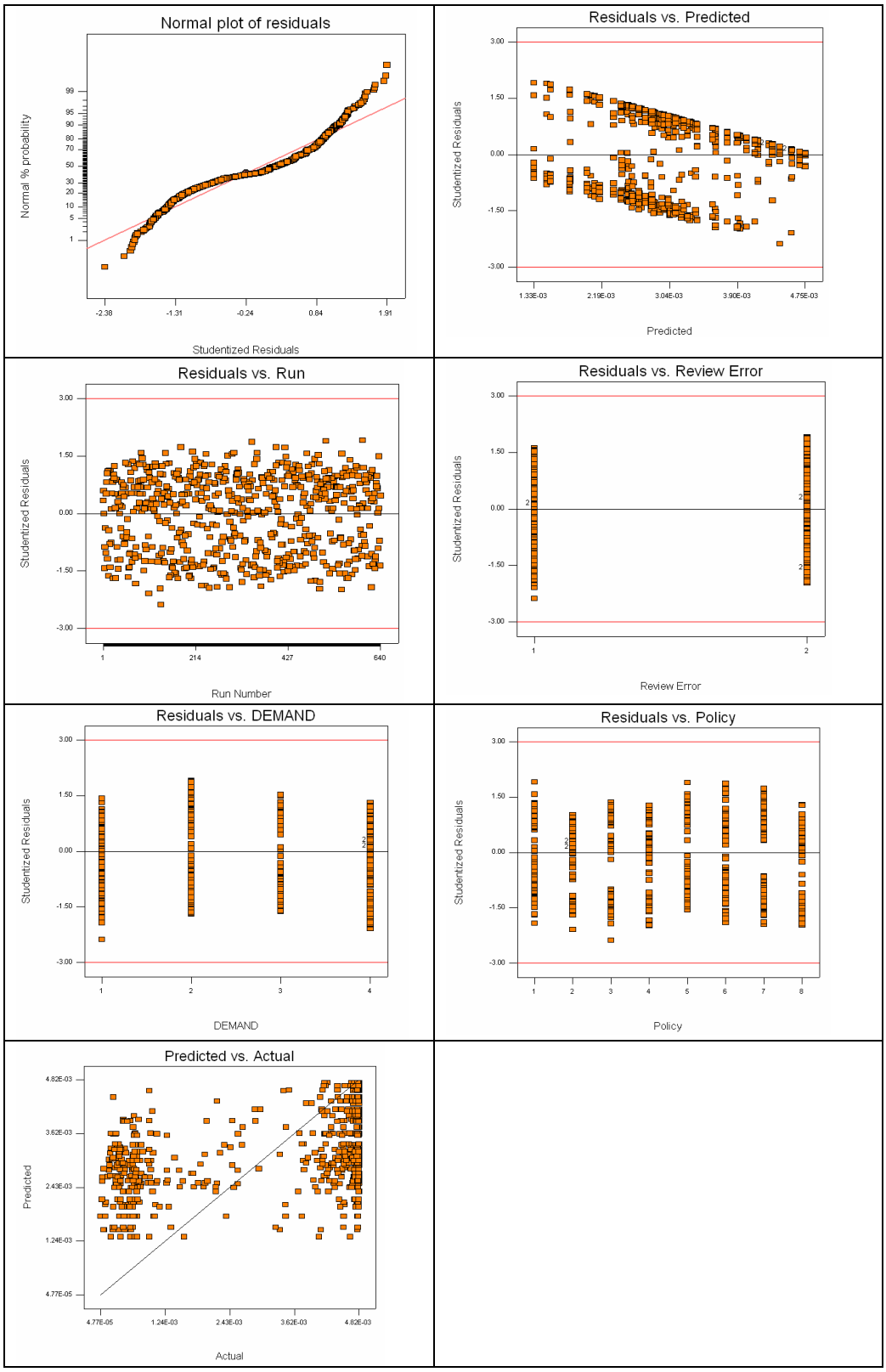


Figure 35: Small workforce vehicle data inventory cost standard deviation ANOVA diagnostics

Appendix F

POLICY	DEMAND	REVIEW	UNIT PRICE	AIRCRAFT					
				OPT Cost	Opt SD	Start Cost	Start SD	Opt Delta	Opt Impact
(.5Q Q P)	Level	Flawed	3.896	2669.665	1597.468	540.167	171.915	$2,129.50	-0.156
(.5Q Q P)	Periodic	Flawed	3.786	4401.243	3418.550	600.314	213.199	$3,800.93	-0.267
(.5Q Q P)	Seasonal	Flawed	2.647	5294.699	3656.507	1764.975	334.851	$3,529.72	-0.356
(.5Q Q P)	Sparse	Flawed	5.755	6888.972	3260.595	65.818	27.682	$6,823.15	-0.022
(Q Q P)	Level	Flawed	3.896	6642.089	2577.626	251.339	121.378	$6,390.75	0.067
(Q Q P)	Periodic	Flawed	3.786	3939.031	2752.137	79.902	19.214	$3,859.13	-0.022
(Q Q P)	Seasonal	Flawed	2.647	7002.384	4441.327	1156.055	78.606	$5,846.33	0.133
(Q Q P)	Sparse	Flawed	5.755	16.974	10.789	17.424	9.739	($0.45)	-0.267
(r Q 1)	Level	Flawed	3.896	15.063	0.189	121.013	4.543	($105.95)	-0.267
(r Q 1)	Periodic	Flawed	3.786	18.928	0.960	191.939	7.225	($173.01)	-0.644
(r Q 1)	Seasonal	Flawed	2.647	10.679	0.037	1144.692	9.686	($1,134.01)	-0.844
(r Q 1)	Sparse	Flawed	5.755	43.246	0.724	43.373	0.762	($0.13)	-0.111
(r Q P)	Level	Flawed	3.896	5351.378	2530.661	540.167	171.915	$4,811.21	-0.022
(r Q P)	Periodic	Flawed	3.786	7307.136	3384.353	600.314	213.199	$6,706.82	-0.111
(r Q P)	Seasonal	Flawed	2.647	6743.261	3920.775	1764.975	334.851	$4,978.29	-0.311
(r Q P)	Sparse	Flawed	5.755	3769.989	2851.068	65.818	27.682	$3,704.17	-0.067
(s S 1)	Level	Flawed	3.896	498.437	0.429	498.437	0.429	$0.00	0.000
(s S 1)	Periodic	Flawed	3.786	1422.418	1.548	1422.418	1.548	$0.00	0.000
(s S 1)	Seasonal	Flawed	2.647	991.654	1.478	991.654	1.478	$0.00	0.000
(s S 1)	Sparse	Flawed	5.755	1357.313	0.281	1355.672	0.285	$1.64	-0.111
(s S P)	Level	Flawed	3.896	509.366	10.277	509.366	10.277	$0.00	0.000
(s S P)	Periodic	Flawed	3.786	2768.479	852.525	1132.063	372.104	$1,636.42	-0.133
(s S P)	Seasonal	Flawed	2.647	1004.466	29.158	1004.466	29.158	$0.00	0.000
(s S P)	Sparse	Flawed	5.755	1360.971	3.996	1361.059	3.996	($0.09)	-0.111
(S-1 S 1)	Level	Flawed	3.896	498.368	0.453	498.368	0.453	$0.00	0.000
(S-1 S 1)	Periodic	Flawed	3.786	1419.623	1.605	1419.623	1.605	$0.00	0.000
(S-1 S 1)	Seasonal	Flawed	2.647	1871.929	0.975	1871.929	0.975	$0.00	0.000
(S-1 S 1)	Sparse	Flawed	5.755	162145.810	0.275	43194.500	0.281	$118,951.31	-11.953
(S-1 S P)	Level	Flawed	3.896	86.498	97.961	86.498	97.961	$0.00	0.000
(S-1 S P)	Periodic	Flawed	3.786	1999.293	1340.499	559.464	55.110	$1,439.83	-0.133
(S-1 S P)	Seasonal	Flawed	2.647	907.662	690.557	907.662	690.557	$0.00	0.000
(S-1 S P)	Sparse	Flawed	5.755	769.514	317.025	823.180	470.558	($53.67)	-0.244
(.5Q Q P)	Level	Perfect	3.896	3745.477	2609.781	589.611	50.718	$3,155.87	0.067
(.5Q Q P)	Periodic	Perfect	3.786	4088.410	2328.366	567.518	156.733	$3,520.89	-0.444
(.5Q Q P)	Seasonal	Perfect	2.647	4327.852	2717.443	1898.395	236.368	$2,429.46	-0.333
(.5Q Q P)	Sparse	Perfect	5.755	4507.628	2928.841	70.403	24.309	$4,437.22	-0.111
(Q Q P)	Level	Perfect	3.896	6677.924	2899.223	221.426	61.466	$6,456.50	0.178
(Q Q P)	Periodic	Perfect	3.786	6880.163	3287.431	81.202	17.548	$6,798.96	0.044
(Q Q P)	Seasonal	Perfect	2.647	8039.502	4292.776	1190.207	117.753	$6,849.30	-0.067
(Q Q P)	Sparse	Perfect	5.755	4443.295	3167.615	24.316	4.765	$4,418.98	0.000
(r Q 1)	Level	Perfect	3.896	15.880	0.202	127.602	5.036	($111.72)	-0.489
(r Q 1)	Periodic	Perfect	3.786	12.976	0.088	239.952	8.173	($226.98)	-0.667
(r Q 1)	Seasonal	Perfect	2.647	14.253	0.139	1186.958	13.778	($1,172.71)	-0.844
(r Q 1)	Sparse	Perfect	5.755	50.753	0.488	51.092	0.446	($0.34)	-0.244

				AIRCRAFT					
POLICY	DEMAND	REVIEW	UNIT PRICE	OPT Cost	Opt SD	Start Cost	Start SD	Opt Delta	Opt Impact
(r Q P)	Level	Perfect	3.896	5180.741	2269.592	589.611	50.718	$4,591.13	0.044
(r Q P)	Periodic	Perfect	3.786	4514.724	2819.793	567.518	156.733	$3,947.21	-0.311
(r Q P)	Seasonal	Perfect	2.647	7171.564	3328.672	1898.395	236.368	$5,273.17	-0.356
(r Q P)	Sparse	Perfect	5.755	4307.442	2341.760	70.403	24.309	$4,237.04	0.000
(s S 1)	Level	Perfect	3.896	51.205	0.608	102.973	3.410	($51.77)	-0.422
(s S 1)	Periodic	Perfect	3.786	204.787	3.570	204.787	3.570	$0.00	0.000
(s S 1)	Seasonal	Perfect	2.647	60.515	1.140	1621.401	11.975	($1,560.89)	-0.489
(s S 1)	Sparse	Perfect	5.755	61.156	0.518	61.053	0.474	$0.10	-0.267
(s S P)	Level	Perfect	3.896	923.105	790.631	111.451	21.145	$811.65	-0.222
(s S P)	Periodic	Perfect	3.786	721.602	229.044	269.309	77.323	$452.29	-0.244
(s S P)	Seasonal	Perfect	2.647	658.811	686.349	1479.898	270.112	($821.09)	-0.400
(s S P)	Sparse	Perfect	5.755	2539.738	1039.928	86.056	24.671	$2,453.68	-0.222
(S-1 S 1)	Level	Perfect	3.896	26.024	0.356	102.094	3.395	($76.07)	-0.422
(S-1 S 1)	Periodic	Perfect	3.786	200.654	4.092	200.654	4.092	$0.00	0.000
(S-1 S 1)	Seasonal	Perfect	2.647	12.772	0.063	1622.695	12.334	($1,609.92)	-0.711
(S-1 S 1)	Sparse	Perfect	5.755	61.202	0.561	60.933	0.467	$0.27	-0.156
(S-1 S P)	Level	Perfect	3.896	1434.375	766.798	113.210	21.296	$1,321.16	-0.200
(S-1 S P)	Periodic	Perfect	3.786	2775.949	1310.622	249.592	44.433	$2,526.36	-0.178
(S-1 S P)	Seasonal	Perfect	2.647	2192.440	1353.278	1404.511	171.349	$787.93	-0.333
(S-1 S P)	Sparse	Perfect	5.755	6007.194	2906.341	87.497	24.438	$5,919.70	-0.022

				VEHICLE, Large workforce					
POLICY	DEMAND	REVIEW	UNIT PRICE	OPT Cost	Opt SD	Start Cost	Start SD	Opt Delta	Opt Impact
(.5Q Q P)	Level	Flawed	146.363	18903.509	4214.9589	212831.822	51616.9035	($193,928.31)	-0.8
(.5Q Q P)	Periodic	Flawed	797.037	45373.367	547.4993	131283.323	21133.1027	($85,909.96)	-0.6
(.5Q Q P)	Seasonal	Flawed	87.726	14078.993	3621.3206	205284.634	54331.6902	($191,205.64)	-1
(.5Q Q P)	Sparse	Flawed	288.963	15285.684	3265.251	93596.171	10422.1361	($78,310.49)	-0.4
(Q Q P)	Level	Flawed	146.363	11608.176	4148.9757	204245.067	54238.3238	($192,636.89)	-0.8
(Q Q P)	Periodic	Flawed	797.037	20888.466	3344.5567	103555.266	14196.0515	($82,666.80)	-0.8
(Q Q P)	Seasonal	Flawed	87.726	11755.587	3986.2812	192007.667	46345.8615	($180,252.08)	-1
(Q Q P)	Sparse	Flawed	288.963	16630.54	4572.7762	90849.004	8443.9194	($74,218.46)	-0.3
(r Q 1)	Level	Flawed	146.363	8102.441	3178.1465	14883.126	6124.7824	($6,780.69)	-0.1
(r Q 1)	Periodic	Flawed	797.037	16990.394	1214.5803	28025.369	1289.0156	($11,034.98)	-0.7
(r Q 1)	Seasonal	Flawed	87.726	9835.222	3046.7085	8669.168	2238.0713	$1,166.05	-0.4
(r Q 1)	Sparse	Flawed	288.963	9056.117	42.8806	9598.588	438.7541	($542.47)	-0.2
(r Q P)	Level	Flawed	146.363	15752.882	4242.8176	32212.727	13376.484	($16,459.85)	-0.4
(r Q P)	Periodic	Flawed	797.037	33353.942	2933.48	34281.087	6288.1299	($927.15)	-0.6
(r Q P)	Seasonal	Flawed	87.726	9045.918	4253.0647	17446.153	7724.2889	($8,400.24)	-0.4
(r Q P)	Sparse	Flawed	288.963	10453.88	1567.058	15248.124	3200.1819	($4,794.24)	-0.1
(s S 1)	Level	Flawed	146.363	29677.972	4308.7067	129538.265	11933.1006	($99,860.29)	-0.9
(s S 1)	Periodic	Flawed	797.037	47945.665	5127.6969	47945.665	5127.6969	$0.00	0
(s S 1)	Seasonal	Flawed	87.726	50582.222	6112.1159	109667.023	16919.1321	($59,084.80)	-0.6
(s S 1)	Sparse	Flawed	288.963	49584.705	1552.854	49689.359	650.0391	($104.65)	-0.3

				VEHICLE, Large workforce					
POLICY	DEMAND	REVIEW	UNIT PRICE	OPT Cost	Opt SD	Start Cost	Start SD	Opt Delta	Opt Impact
(s S P)	Level	Flawed	146.363	56438.6	5826.5779	131122.849	27847.4815	($74,684.25)	-0.6
(s S P)	Periodic	Flawed	797.037	31937.251	6438.2543	46838.264	10432.0304	($14,901.01)	-0.2
(s S P)	Seasonal	Flawed	87.726	62962.163	12983.2372	111242.666	24853.1554	($48,280.50)	-0.4
(s S P)	Sparse	Flawed	288.963	30810.249	2686.1667	53437.332	5554.2397	($22,627.08)	-0.4
(S-1 S 1)	Level	Flawed	146.363	11828.842	3598.2785	312445.292	12266.0121	($300,616.45)	-0.9
(S-1 S 1)	Periodic	Flawed	797.037	70286.649	2241.5229	138507.466	5279.5316	($68,220.82)	-0.7
(S-1 S 1)	Seasonal	Flawed	87.726	85454.45	6079.9339	258455.335	16134.4722	($173,000.89)	-0.6
(S-1 S 1)	Sparse	Flawed	288.963	$28,637.75	1782.8728	$105,763.28	4104.3384	($77,125.53)	-0.6
(S-1 S P)	Level	Flawed	146.363	27866.522	5754.7048	291340.998	70321.2836	($263,474.48)	-0.9
(S-1 S P)	Periodic	Flawed	797.037	57130.354	3203.9898	129546.539	19746.924	($72,416.19)	-0.7
(S-1 S P)	Seasonal	Flawed	87.726	50881.885	9252.1358	252921.103	62883.2301	($202,039.22)	-0.6
(S-1 S P)	Sparse	Flawed	288.963	44016.351	3656.9885	109156.354	8154.5431	($65,140.00)	-0.2
(.5Q Q P)	Level	Perfect	146.363	37165.653	4311.8165	211269.614	51295.6305	($174,103.96)	-0.8
(.5Q Q P)	Periodic	Perfect	797.037	45602.968	393.1519	129591.325	23949.3823	($83,988.36)	-0.6
(.5Q Q P)	Seasonal	Perfect	87.726	14008.461	3099.9361	204784.672	54586.1739	($190,776.21)	-1
(.5Q Q P)	Sparse	Perfect	288.963	10938.851	1030.7991	94617.817	9109.4457	($83,678.97)	-0.5
(Q Q P)	Level	Perfect	146.363	18501.769	4626.5936	203617.817	48355.8569	($185,116.05)	-0.8
(Q Q P)	Periodic	Perfect	797.037	16792.472	54.3804	105510.807	14954.8776	($88,718.34)	-0.8
(Q Q P)	Seasonal	Perfect	87.726	11311.516	4040.6025	194892.847	46786.6437	($183,581.33)	-1
(Q Q P)	Sparse	Perfect	288.963	21774.028	5756.0753	90325.793	9465.3732	($68,551.77)	-0.2
(r Q 1)	Level	Perfect	146.363	12962.405	3201.3609	24882.879	5310.2021	($11,920.47)	-0.7
(r Q 1)	Periodic	Perfect	797.037	16738.592	62.738	29568.602	817.3889	($12,830.01)	-0.9
(r Q 1)	Seasonal	Perfect	87.726	9802.171	2260.4698	15508.69	2869.501	($5,706.52)	0
(r Q 1)	Sparse	Perfect	288.963	5990.132	132.8648	11032.557	73.2342	($5,042.43)	-0.2
(r Q P)	Level	Perfect	146.363	7765.802	3509.3247	37548.882	12568.9892	($29,783.08)	-0.8
(r Q P)	Periodic	Perfect	797.037	36958.955	3279.7735	35559.587	5458.6231	$1,399.37	-0.4
(r Q P)	Seasonal	Perfect	87.726	9567.677	3292.1958	20596.158	7068.7743	($11,028.48)	-0.4
(r Q P)	Sparse	Perfect	288.963	13114.984	3376.5023	15207.03	1249.9955	($2,092.05)	-0.1
(s S 1)	Level	Perfect	146.363	35590.743	4237.3303	135748.478	17185.9776	($100,157.74)	-0.9
(s S 1)	Periodic	Perfect	797.037	57649.366	4067.6648	57649.366	4067.6648	$0.00	0
(s S 1)	Seasonal	Perfect	87.726	42689.15	6232.7892	112744.709	16882.7133	($70,055.56)	-0.4
(s S 1)	Sparse	Perfect	288.963	28191.586	406.9773	42291.539	1816.2701	($14,099.95)	-0.3
(s S P)	Level	Perfect	146.363	41231.728	4139.8557	138182.396	28848.1373	($96,950.67)	-0.9
(s S P)	Periodic	Perfect	797.037	37921.159	6362.89	50531.211	11589.5538	($12,610.05)	-0.4
(s S P)	Seasonal	Perfect	87.726	40155.938	3804.8787	115609.094	27454.0944	($75,453.16)	-1
(s S P)	Sparse	Perfect	288.963	29381.184	4201.661	41921.333	7948.341	($12,540.15)	-0.4
(S-1 S 1)	Level	Perfect	146.363	8969.713	2841.4603	310720.425	15139.1235	($301,750.71)	-0.9
(S-1 S 1)	Periodic	Perfect	797.037	44270.502	969.8781	136521.497	5843.8503	($92,251.00)	-0.7
(S-1 S 1)	Seasonal	Perfect	87.726	8174.762	2471.8051	253258.066	12099.3109	($245,083.30)	-1
(S-1 S 1)	Sparse	Perfect	288.963	7981.583	22.9679	105045.585	3959.3386	($97,064.00)	-0.8
(S-1 S P)	Level	Perfect	146.363	28809.974	6718.3157	294924.185	69577.3805	($266,114.21)	-0.9
(S-1 S P)	Periodic	Perfect	797.037	46931.226	3301.0407	127661.06	19366.5006	($80,729.83)	-0.6
(S-1 S P)	Seasonal	Perfect	87.726	35741.813	8926.371	251366.201	65017.1078	($215,624.39)	-1
(S-1 S P)	Sparse	Perfect	288.963	17509.472	2485.8422	106187.934	10317.2567	($88,678.46)	-0.4

				Vehicle, Small workforce					
POLICY	DEMAND	ERROR	UNIT PRICE	OPT Cost	Opt SD	Start Cost	Start SD	Opt Delta	Opt Impact
(r Q P)	Periodic	Flawed	797.037	5586.403	739.87	7605.716	838.583	($2,019.31)	0.2
(r Q P)	Seasonal	Flawed	87.726	5093.514	1258.8	6452.375	1407.35	($1,358.86)	-0.4
(r Q P)	Level	Flawed	146.363	6706.332	1003.5	9176.914	1848.29	($2,470.58)	-0.5
(r Q P)	Sparse	Flawed	288.963	4075.029	820.97	3346.819	508.98	$728.21	0.1
(r Q 1)	Periodic	Flawed	797.037	5396.052	34.781	6188.554	196.444	($792.50)	-0.5
(r Q 1)	Seasonal	Flawed	87.726	3130.984	452.59	3130.984	452.594	$0.00	0
(r Q 1)	Level	Flawed	146.363	3909.781	405.97	5948.689	827.091	($2,038.91)	-0.1
(r Q 1)	Sparse	Flawed	288.963	2002.281	11.277	2338.984	69.2624	($336.70)	-0.2
(Q Q P)	Periodic	Flawed	797.037	4195.211	136.83	19348.408	1798.3	($15,153.20)	-0.8
(Q Q P)	Seasonal	Flawed	87.726	9791.632	967.78	52412.352	6004.47	($42,620.72)	-0.8
(Q Q P)	Level	Flawed	146.363	6290.297	844.37	52773.193	7282.32	($46,482.90)	-0.9
(Q Q P)	Sparse	Flawed	288.963	4824.183	1079.4	17654.738	1075.16	($12,830.56)	-0.3
(.5Q Q P)	Periodic	Flawed	797.037	7293.874	30.558	24045.399	2724.98	($16,751.53)	-0.8
(.5Q Q P)	Seasonal	Flawed	87.726	14603.51	1413.7	56663.93	7357.27	($42,060.42)	-1
(.5Q Q P)	Level	Flawed	146.363	17103.03	978.96	56637.629	6966.47	($39,534.60)	-0.9
(.5Q Q P)	Sparse	Flawed	288.963	2620.722	235.07	18211.275	1365.74	($15,590.55)	-0.5
(s S P)	Periodic	Flawed	797.037	8201.556	918.63	9868.176	1405.24	($1,666.62)	-0.4
(s S P)	Seasonal	Flawed	87.726	26173.99	2919.3	34736.773	3597.66	($8,562.79)	-0.4
(s S P)	Level	Flawed	146.363	23884.04	614.79	42640.112	3815.63	($18,756.07)	-0.7
(s S P)	Sparse	Flawed	288.963	8173.984	1058.2	11343.634	661.572	($3,169.65)	0.1
(s S 1)	Periodic	Flawed	797.037	9871.586	613.77	9871.586	613.774	$0.00	0
(s S 1)	Seasonal	Flawed	87.726	25452.97	1532.2	34854.109	2499.58	($9,401.14)	-0.4
(s S 1)	Level	Flawed	146.363	24064.7	802.57	41649.859	1886.55	($17,585.16)	-0.7
(s S 1)	Sparse	Flawed	288.963	8261.217	161.37	10939.53	63.8126	($2,678.31)	-0.2
(S-1 S P)	Periodic	Flawed	797.037	11948.9	645.18	23818.287	2559.91	($11,869.38)	-0.7
(S-1 S P)	Seasonal	Flawed	87.726	21797.08	3571.6	65021.25	8381.63	($43,224.17)	-0.6
(S-1 S P)	Level	Flawed	146.363	15086.26	1729.9	74351.677	9271.72	($59,265.42)	-0.9
(S-1 S P)	Sparse	Flawed	288.963	8393.508	794.83	20499.284	1130.57	($12,105.78)	-0.3
(S-1 S 1)	Periodic	Flawed	797.037	17974.62	617.87	24687.86	614.635	($6,713.24)	-0.5
(S-1 S 1)	Seasonal	Flawed	87.726	17412.74	727.43	64890.081	2115.93	($47,477.34)	-0.6
(S-1 S 1)	Level	Flawed	146.363	5019.412	449.86	76855.689	1600.73	($71,836.28)	-0.9
(S-1 S 1)	Sparse	Flawed	288.963	11379.41	349.84	20363.068	470.714	($8,983.66)	-0.5
(r Q P)	Periodic	Perfect	797.037	7715.263	426.41	7805.152	726.942	($89.89)	-0.4
(r Q P)	Seasonal	Perfect	87.726	4235.328	473.58	7994.953	1108.99	($3,759.63)	-0.1
(r Q P)	Level	Perfect	146.363	8949.217	925.04	11067.211	1920.39	($2,117.99)	-0.3
(r Q P)	Sparse	Perfect	288.963	2890.644	498.53	3840.413	247.87	($949.77)	-0.3
(r Q 1)	Periodic	Perfect	797.037	4438.957	8.1298	6516.175	112.949	($2,077.22)	-0.3
(r Q 1)	Seasonal	Perfect	87.726	3845.014	221.72	6075.458	350.785	($2,230.44)	-0.4
(r Q 1)	Level	Perfect	146.363	6853.483	411.47	8523.762	768.623	($1,670.28)	-0.1
(r Q 1)	Sparse	Perfect	288.963	1814.622	93.998	2851.505	13.2981	($1,036.88)	-0.4
(Q Q P)	Periodic	Perfect	797.037	5074.536	333.3	19624.838	1911.77	($14,550.30)	-0.8
(Q Q P)	Seasonal	Perfect	87.726	4708.091	820.36	52813.005	6129.45	($48,104.91)	-1
(Q Q P)	Level	Perfect	146.363	7376.446	1028.5	52470.872	6471.56	($45,094.43)	-0.9
(Q Q P)	Sparse	Perfect	288.963	4009.107	792.22	17634.318	1214.11	($13,625.21)	-0.5
(.5Q Q P)	Periodic	Perfect	797.037	7410.739	10.519	23846.044	3041.74	($16,435.31)	-0.8
(.5Q Q P)	Seasonal	Perfect	87.726	5285.662	405.05	56653.397	7409.68	($51,367.74)	-1
(.5Q Q P)	Level	Perfect	146.363	20728.23	817.41	56406.957	6862.02	($35,678.73)	-0.9
(.5Q Q P)	Sparse	Perfect	288.963	3310.82	304.57	18309.532	1186.36	($14,998.71)	-0.5

				Vehicle, Small workforce					
POLICY	DEMAND	ERROR	UNIT PRICE	OPT Cost	Opt SD	Start Cost	Start SD	Opt Delta	Opt Impact
(s S P)	Periodic	Perfect	797.037	8608.939	578.88	11452.218	1455.93	($2,843.28)	0
(s S P)	Seasonal	Perfect	87.726	23847.78	1527	36298.841	3873.52	($12,451.07)	-0.6
(s S P)	Level	Perfect	146.363	18420.72	1561	41669.954	3901.93	($23,249.24)	-0.9
(s S P)	Sparse	Perfect	288.963	8476.375	1061.8	10107.259	994.168	($1,630.88)	-0.5
(s S 1)	Periodic	Perfect	797.037	12398.31	551.6	12398.314	551.604	$0.00	0
(s S 1)	Seasonal	Perfect	87.726	22657.18	1130.6	36088.118	2157.99	($13,430.94)	-0.6
(s S 1)	Level	Perfect	146.363	13257.67	486.54	41946.343	2247.86	($28,688.67)	-0.9
(s S 1)	Sparse	Perfect	288.963	7577.138	252.65	9931.128	388.362	($2,353.99)	-0.4
(S-1 S P)	Periodic	Perfect	797.037	10419.17	828.52	23604	2422.32	($13,184.83)	-0.6
(S-1 S P)	Seasonal	Perfect	87.726	11882.62	1955.6	64704.176	8615.89	($52,821.56)	-0.8
(S-1 S P)	Level	Perfect	146.363	16820.28	2138.5	74855.315	9154.65	($58,035.04)	-0.9
(S-1 S P)	Sparse	Perfect	288.963	5462.601	409.58	20114.758	1347.33	($14,652.16)	-0.5
(S-1 S 1)	Periodic	Perfect	797.037	8269.311	86.676	24482.84	749.591	($16,213.53)	-0.9
(S-1 S 1)	Seasonal	Perfect	87.726	3404.206	306.44	64131.908	1522.91	($60,727.70)	-1
(S-1 S 1)	Level	Perfect	146.363	5579.799	445.22	76563.601	2536.77	($70,983.80)	-0.9
(S-1 S 1)	Sparse	Perfect	288.963	3002.923	13.074	20225.846	502.721	($17,222.92)	-0.8

Index

T

W