NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

SIMULATED OPERATING CONCEPTS FOR WIOM IMPLEMENTATION

by

Sean M. Teter

March 2018

Thesis Advisor: Emily Craparo
Second Reader: Javier Salmeron

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE March 2018	3. REPORT TYPE AND DATES COVERED Master's thesis

4. TITLE AND SUBTITLE SIMULATED OPERATING CONCEPTS FOR WIOM IMPLEMENTATION	5. FUNDING NUMBERS
6. AUTHOR(S) Sean M. Teter	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB number ____N/A____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT (maximum 200 words)

Naval Supply Systems Command Weapon Systems Support (NAVSUP WSS) serves as the Navy's inventory control point, managing approximately 375,000 line items. Constrained by funding, NAVSUP WSS uses the Wholesale Inventory Optimization Model (WIOM), a mixed-integer linear program developed by Naval Postgraduate School faculty, to maximize customer service. Since demand distributions for different parts change over time, NAVSUP WSS updates the inputs to WIOM and reruns it quarterly. However, large changes to the solution create an administrative burden. To deal with this problem, referred to as churn, WIOM has a persistence parameter that can discourage change from one run to the next, but it is inherently at odds with customer service performance.

This thesis presents a new model, the Comparative Optimized Results Simulation (CORS). Using CORS, the thesis explores the system's performance under different settings of the persistence parameter and different periodicities of running WIOM. The thesis finds that periodicities greater than quarterly significantly degrade customer service. Additionally, the thesis finds that increasing the persistence parameter dramatically improves churn while only marginally degrading customer service.

14. SUBJECT TERMS inventory management; discrete event simulation; wholesale inventory optimization model	15. NUMBER OF PAGES 79
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

SIMULATED OPERATING CONCEPTS FOR WIOM IMPLEMENTATION

Sean M. Teter
Lieutenant Commander, United States Navy
B.A., Florida State University, 2005
B.A., Florida State University, 2005

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
March 2018

Approved by: Dr. Emily Craparo
Thesis Advisor

Dr. Javier Salmeron
Second Reader

Dr. Patricia Jacobs
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Naval Supply Systems Command Weapon Systems Support (NAVSUP WSS) serves as the Navy's inventory control point, managing approximately 375,000 line items. Constrained by funding, NAVSUP WSS uses the Wholesale Inventory Optimization Model (WIOM), a mixed-integer linear program developed by Naval Postgraduate School faculty, to maximize customer service. Since demand distributions for different parts change over time, NAVSUP WSS updates the inputs to WIOM and reruns it quarterly. However, large changes to the solution create an administrative burden. To deal with this problem, referred to as churn, WIOM has a persistence parameter that can discourage change from one run to the next, but it is inherently at odds with customer service performance.

This thesis presents a new model, the Comparative Optimized Results Simulation (CORS). Using CORS, the thesis explores the system's performance under different settings of the persistence parameter and different periodicities of running WIOM. The thesis finds that periodicities greater than quarterly significantly degrade customer service. Additionally, the thesis finds that increasing the persistence parameter dramatically improves churn while only marginally degrading customer service.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

CORS	Comparative Optimized Results Simulation
ERP	Navy Enterprise Resource Planning
FY	fiscal year
IP	inventory position
LSSI	level setting strategy indicator
LT	lead time
NAVSUP WSS	Naval Supply Systems Command, Weapon System Support
NIIN	national item identification number
Q	order quantity
Q_O/H	quantity on-hand
ROP	reorder point
SPO	Service Planning and Optimization
WIOM	Wholesale Inventory Optimization Model

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Naval Supply Systems Command, Weapon Systems Support (NAVSUP WSS) serves as the main inventory control point for the Navy, managing approximately 375,000 unique line items. NAVSUP WSS strives to maintain the best possible material support to the fleet by effectively managing the Navy's wholesale inventory. Constrained by budget, it does this by managing when it orders material.

To optimize this support, NAVSUP WSS uses the Wholesale Inventory Optimization Model (WIOM), a tool developed by Naval Postgraduate School faculty (Salmeron and Craparo 2017). WIOM strives to maximize a function closely related to *fill rate*, which is a standard measure of customer support, while staying within budgetary constraints. NAVSUP WSS runs WIOM once a quarter. In his 2016 NPS thesis, Lieutenant Commander Geoffrey Roth used a simulation study to show that WIOM performed better than NAVSUP WSS's legacy optimization model. Since this research, NAVSUP WSS identified desirable new features for WIOM. One of these new features, persistence, was added to WIOM in order to preserve legacy values from previous solutions. This reduces what is known as churn: the change in solution from one model run to the next.

The reduction of churn is beneficial for NAVSUP WSS from an administrative perspective. However, enforcing persistence may also reduce fill rate performance and support to the fleet. This thesis develops the Comparative Optimized Results Simulation (CORS) in order to test wholesale inventory performance. CORS is a discrete event simulation that uses 4.5 years of historic demand data provided by NAVSUP WSS as input and allows multiple runs of WIOM during the simulation period. This is fundamentally different from the previous simulation study, which used one WIOM run and stochastic demand arrivals with the assumption that the underlying demand patterns were unchanging.

Using CORS, the thesis tests the effects of modifying two variables: persistence parameter and periodicity of running WIOM. We consider persistence parameter settings at four levels we define as none, low, medium and high. We consider quarterly, semi-

annual, and annual periodicities. We create 15 different combinations of periodicities and persistence parameter settings and use CORS to test inventory system performance in terms of simulated fill rate under these settings.

The thesis gains several insights from the experimental results. First, fill rates between poor- and high-performing designs take time to diverge. An excellent and poor design take at least six months before a difference in performance is noted. Next, we conclude that designs with quarterly periodicities clearly outperform semi-annual and annual periodicities. WIOM solutions appear to "expire" as time passes and underlying demand patterns of the system change. Finally, we determine that churn can be drastically reduced without sacrificing system performance. In our experiments we are able to reduce the churn by 99% without practically significant degradation in fill rate. In fact, we are unable to substantially reduce fill rate performance by increasing the persistence parameter. We find that increasing the persistence parameter has a decreasing marginal effect on churn, and above a certain level has no further effect and a minimum churn is reached. In this case the minimum level of churn reached by WIOM was not constraining enough to cause a reduction in fill rate performance. We do not, however, conclude that this is the case generally, and further research is warranted.

References

Roth G (2016) A simulation of alternative for wholesale inventory replenishment. Master's thesis, Operations Research Department, Naval Postgraduate School, Monterey, CA. https://calhoun.nps.edu/handle/10945/48587.

Salmeron J, Craparo E (2017) *Wholesale Inventory Optimization Model.* Naval Postgraduate School, Monterey, CA.

ACKNOWLEDGMENTS

Completing this academic program has been a humbling experience. Having good people around me has helped immensely. I'd like to thank all of the fellow students in my cohort who weathered the storm alongside me, and all the professors who guided us through to calmer waters on the other side. I couldn't have asked for a better group of either.

Completing this thesis proved to be even more humbling. Immense thanks go to my advisor, Emily Craparo, for her constant support, encouragement, and guidance. Further thanks to my second reader, Javier Salmeron, who refused to accept any less than my best work and made me better in the process.

I would never be here in the first place if it weren't for my family. My parents have been unfailing in support for every endeavor I have ever taken and have been my biggest cheerleaders. My big sister, Caitlin, has been a constant role model I have looked up to for as long as I can remember. While I was initially (and vocally) disappointed that my little sister, Claire, wasn't a boy, I guess she's okay too.

Of course, no thanks or words in the world can express my gratitude to my wife, Judy. Her love, encouragement, and strength through this process made it possible. While there's no dedication section in this thesis, I choose to dedicate it nonetheless:

For my wife and daughter, the loves of my life.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

Naval Supply Systems Command, Weapon Systems Support (NAVSUP WSS) serves as the main inventory control point for the Navy. The command manages over 375,000 unique line items (NAVSUP 2018) used in the repair of ships, submarines, Navy and Marine Corps aircraft, and associated weapons systems. The effective management of this supply chain is essential in maintaining readiness of the fleet to operate and conduct combat operations around the world.

Like any organization, NAVSUP WSS has a limited set of resources with which to conduct its operations. The biggest constraint is financial. Given limited budgetary means, NAVSUP WSS strives to maximize support to the warfighter. The predominant metric used to measure customer support is *fill rate*. When NAVSUP WSS receives a requisition, one of two things can happen. Either the requisition is filled immediately with stock on hand, or the requisition is backordered. The fill rate metric shows the relationship between the number of requisitions filled immediately on receipt and the number of requisitions that are backordered. Fill rate is defined mathematically as follows:

$$\textit{Fill rate} = \textit{Requisitions Filled} / \textit{Requisitions Received.}$$

For example, if 50 requisitions were received in a given period, and 43 of them were filled and 7 were backordered, then a fill rate of .86 or 86% was achieved for this period. The above calculation can be applied to a specific item or to a group of items. When it is applied to a group of items, it can be done in one of two ways. First, the fill rate can be calculated as an average of all the individual item fill rates. Or, the fill rate can be calculated with the above equation without regard to what the particular item is. This is also called demand weighting, because it is equivalent to a weighted average of item fill rates, weighted according to the demands of the individual items. In this thesis, we use demand weighted fill rate unless specifically noted otherwise.

In the past NAVSUP WSS used commercially-developed optimization software to maximize their achieved fill rate given their budget constraints. Developed by MCA

Solutions, the Service Planning and Optimization (SPO) was effective but had shortcomings. First, it was a "black box" to the users at NAVSUP WSS, who did not have access to the models and algorithms SPO used to develop its solutions. SPO did not have the ability to accept budget as a constraint. Therefore, NAVSUP WSS had to run SPO iteratively, adjusting a fill rate constraint until a satisfactory budget figure was reached. Additionally, SPO was expensive, costing around $800,000 per year in licensing fees.

In order to replace SPO with a better-functioning optimization tool at reduced cost, Naval Postgraduate School faculty developed the Wholesale Inventory Optimization Model (WIOM) (Salmeron and Craparo 2017). WIOM is a mixed-integer linear program designed to maximize a function closely related to fill rate, for the wholesale inventory managed by NAVSUP WSS. In his 2016 thesis, Lieutenant Commander Geoffrey Roth used simulation modeling to conclude that WIOM 3.51 was in fact superior to SPO in maximizing fill rates. NAVSUP WSS sunset SPO and began using WIOM in April of 2017.

While WIOM performs well compared to SPO, NAVSUP WSS identified further features they would like to be incorporated into WIOM. First, WIOM 3.51 did not use demand weighting. Instead, it had two settings that could be used. First, WIOM could treat each National Item Identification Number (NIIN) equally. This is not desirable because it ignores the relative importance of NIINs with high demand. Alternatively, WIOM could give preferential treatment to NIINs that were assigned to specific groups called level-setting strategy indicators (LSSIs). By assigning high-demand NIINs to a certain LSSI and then assigning that LSSI a high weight, NAVSUP WSS could mitigate the demand weighting issue. Additionally, NAVSUP WSS could use a series of business rules to create low-demand cutoff points, choosing to leave very low demand NIINs out of the optimization altogether. In order to address this concern, WIOM was revised to use demand weighting, and incorporated this change into the WIOM 4.1 release.

NAVSUP WSS has an additional concern with WIOM (and SPO before it): churn. Churn is the change between solutions from one model run to the next. NAVSUP WSS runs the optimization model once every quarter. In the three months between model runs, the number of requisitions received changes the demand parameters that feed into WIOM. Subsequently, the optimization problems are quite different and considerably differing

solutions are possible. Indeed, if multiple optimal (or near-optimal) solutions exist, churn may occur even in the absence of changes to the input data. This churn creates an administrative burden in contracting and can reduce senior leadership's confidence in optimization efforts. To deal with the churn problem, Salmeron and Craparo (2017) included a term in WIOM's objective function that calculates a churn penalty. This term contains two penalty parameters. One is indexed by NIIN, allowing the user to adjust the relative importance of each NIIN within the churn term. The other is a global persistence parameter that reflects the overall importance of the churn term. This thesis focuses on the global persistence parameter; for simplicity we use the term "persistence parameter" hereafter. The persistence parameter rewards a solution for maintaining legacy values from one model run to the next. The parameter is not an on/off switch; rather, it is a continuous parameter that can be set from zero to an arbitrarily large number. At zero, the persistence parameter is "off." As the parameter increases, the model more strongly prefers to retain incumbent solutions. Additionally, there is an inherent tradeoff between churn reduction and achieved fill rate. The higher the persistence parameter, the less important fill rate becomes in the objective function.

B. LITERATURE REVIEW

1. Inventory Management

Wholesale inventory management is concerned with finding strategies to meet demand requirements from customers at an acceptable service level and an acceptable cost level. Many different models have been proposed, but the two we will discuss are the order-point, order-quantity (s,Q) model and the classic inventory model.

Order-point, order-quantity models are discussed in Silver et al. (1998). In an (s,Q) system, two parameters are used to make decisions on stock replenishment. The first is the reorder point, s. As an item's stock level decreases, a reorder is triggered once the item's inventory position decreases to the level of the reorder point. Inventory position is defined as the quantity on hand plus the quantity on order minus the quantity in a backordered status (i.e., owed to customers). The second parameter is the order quantity Q. This is the quantity of material ordered every time there is a reorder. When a reorder is placed, the

time it takes for this order to arrive is known as the lead time. A key feature of an (s,Q) system is that each reorder is triggered by a low inventory position, not low inventory on hand. This prevents the system from placing extra orders when there is already an order due-in that will replenish stock sufficiently. Silver et al. provide an analogy: "A good example of ordering on the basis of inventory position is the way a person takes aspirin to relieve a headache. After taking two aspirin, it is not necessary to take two more every five minutes until the headache goes away. Rather, it is understood that the relief is 'on order'—aspirin operates with a delay" (Silver et al. 1998).

WIOM uses the (s,Q) system to model NAVSUP WSS's wholesale inventory. However, NAVSUP WSS only determines reorder points. The quantity of the reorders is decided by Navy Enterprise Resource Planning (ERP), and is treated as input by NAVSUP WSS, who then strives to maximize effectiveness by deciding on appropriate reorder points.

A special case of the (s,Q) system is the classical inventory model discussed in Tersine (1994). The classical inventory model uses an (s,Q) system but with a very rigid set of assumptions. Among other things that are not relevant to our purposes, the classical inventory model assumes the following:

- Deterministic and constant demand
- Constant deterministic lead time
- Reorders arrive as a whole lot of size Q
- Backorders are not allowed, since demand and leadtime are constant they are avoided

The resulting system creates a characteristic saw-tooth pattern as shown in Figure 1. This inventory model is used primarily as a means to estimate an order quantity that minimizes cost, known as the economic order quantity. Since NAVSUP WSS treats the order quantity as a given input from ERP, we are not concerned with that aspect of the model. However, the model has some unique qualities that we will use when establishing initial conditions for our simulation. Specifically, a result of the model is that the average

amount of inventory on hand is equal to Q/2. Furthermore, the inventory on hand at any given time is distributed uniformly from zero to Q.

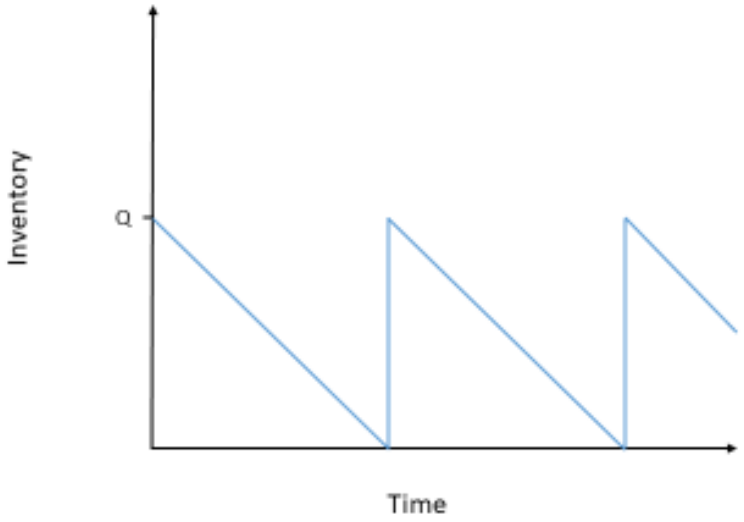

Figure 1. Classical Inventory Model

2. Discrete Event Simulation

Discrete event simulation is addressed in detail in Law (2015). Discrete event simulations are those that advance time from one discrete event to the next. These events may change the state of the system being represented, and the system cannot change during the time between events. Law presents several important definitions to understand such a simulation:

System state: The collection of state variables necessary to describe the system at a particular time;

Simulation Clock: A variable giving the current value of simulated time;

Event List: A list containing the next time when each type of event will occur;

Initialization Routine: A subprogram to initialize the simulation model at time 0;

> Event Routine: A subprogram that updates the system state when a particular type of even occurs (there is one event routing for each event type). (Law 2015)

This thesis develops a simulation using this next-event time advance principle. Events in the system are arranged in time in an event list. The simulated time moves forward from one event to the next according to the events' arrangement in time. The current event is evaluated, state changes to the system are made as necessary, and the simulation moves to the next event in time while the simulation clock is updated.

3. Previous WIOM Simulation Study

In his 2016 thesis, Geoffrey Roth conducted a comparative simulation study between three different optimization methods: simple calculation (a heuristic), SPO, and WIOM. Using a discrete event simulation and testing across five types of material, Roth concluded that WIOM was the best performing of these three alternatives. However, Roth's simulation relies on several strong assumptions:

- NIIN demand probability distributions are known and unchanging through time
- NIIN demands arrive in quantities of one only
- Demands are uncorrelated between NIINs

In addition to these assumptions, the simulation models a lengthy warm-up period of 400,000 days to reach steady state. Due to these assumptions and warm-up period, Roth's simulation would be ineffective to try to model short-term performance of the system with frequent WIOM runs and changes in estimated demand distributions every quarter.

C. OBJECTIVES

The thesis creates a discrete event simulation that uses historical requisitions as input and requires no warm-up period. We call this simulation the Comparative Optimized Results Simulation (CORS). By using historical data and not requiring a warm-up period,

CORS allows for multiple runs of WIOM during the test period. This thesis conducts a series of experiments using the simulation and analyzes the output in order to:

- Gain insight into the relative tradeoff between churn and fill rate using differing settings for the persistence parameter.
- Gain insight into the effect of WIOM periodicity on fill rate.

D. SCOPE, LIMITATIONS, AND ASSUMPTIONS

During the course of the research, we restrict ourselves to looking at the impact of running WIOM at differing periodicities and with differing persistence parameters. In practice, NAVSUP WSS has historically used a set of business rules to help it overcome limitations in SPO. These business rules include mandating minimum and maximum reorder points for some NIINs, which restrict the range of solutions that SPO can use. Additionally, NAVSUP WSS would not input NIINs with exceptionally low demand into SPO. While NAVSUP WSS may choose to continue using these business rules, the current version of WIOM accounts for churn by use of the persistence parameter and accounts for low demand by using demand weighting. Therefore, no additional business rules will be used in this study.

While exploring differing concepts of operations for NAVSUP WSS, we do not explore all possible periodicities. Running WIOM and implementing its solution is administratively burdensome, and organizationally NAVSUP WSS wants to maintain a normal battle rhythm (Ellis et al. 2017). For this reason, we assume that WIOM can only be run quarterly, semiannually, or annually.

The thesis is limited to non-nuclear consumable material. Modeling repairable material is more complex and not addressed in this study.

CORS does not attempt to model all aspects of inventory management. Therefore, while the model delivers insight into performance, it only does so relatively. That is to say, we are only comparing between simulations and claiming which operating condition performed better. A simulation output is not an absolute prediction of how the system would have performed in real life. For example, say the simulations of concept of

operations *a* and concept of operations *b* give overall fill rates of 75% and 70%. In this case, we assert that *a* performed better than *b*. But, we do not make the assertion that the actual fill rate would have been 75% had *a* been in place in real life.

Using deterministic demand gives great flexibility to explore the effects of different concepts of operations that a long term steady state simulation does not. However, by using deterministic demand we are essentially restricted to one data point and a trace simulation. Thus, our conclusions are inherently limited. We can say that one concept of operations performed better than another in the simulation, but only for the given set of demands. There is no basis to assert with confidence that the same would be true for a different set of demands from the same underlying demand distributions.

II. DATA AND METHODOLOGY

A. DATA

In order to run CORS, we need two main sets of data. First, CORS needs optimization output from WIOM (or SPO). Second, CORS needs historical requisition data. To obtain these data we reorganize data received from NAVSUP WSS, which was provided in four forms for fiscal year (FY) 2013 through FY2017 (1 Oct 2012–30 Sep 2017) (Ellis 2017).

The simplest set of data provided is budget figures. NAVSUP WSS provided the historical budgetary constraint placed on each class of material for each quarter of the period of interest (Motter 2017). Instead of using the budget data as provided, the mean of the budget across the period of interest is taken and this constant budget is used throughout. This is because there is an instance when the historical budget changed in the middle of the fiscal year. As will be discussed later, some of our experimental designs will only run WIOM annually. Using the mean allows the experiments to be comparable for different periodicities. Additionally, there is no serious tradeoff by taking the mean since we are not comparing simulation performance to actual performance in our experiments. We only require that budget information be representative.

The set of data provided includes historical requisitions (Ellis 2017). The requisition data are a record of all demands that NAVSUP WSS received during the time period. Each line item in the data represents a single requisition received from the fleet and has 75 data elements recorded. However, most of the data elements are not relevant to running CORS, and we focus on only a few elements. For each requisition, we need to know the NIIN, if the NIIN is a repairable or consumable material, if the NIIN is for aviation or maritime material, if the NIIN is nuclear material, if the requisition was filled, and the Julian date of the requisition. Of note, the Julian date does not represent the date a requisition is received by NAVSUP WSS. Rather, it is part of the template of a requisition number that is assigned by the originating activity when the requisition is created. However, a number of factors could lead to a delay in the requisition being transmitted

after it has been created. In the absence of better information, however, we assume that the Julian date represents the date the requisition is received by NAVSUP WSS.

The next set of data provided consists of historical candidates files (Ellis 2017). These files contain information for all the NIINs that were input into SPO for each quarter. The files contain 20 data elements for each NIIN that are necessary to run WIOM. These files are ready to input into WIOM. However, some issues with the data prevent their unaltered use. First, these files are not available for the entire period. Files are only available for the quarters between and including April 2014 and July 2017. Secondly, relatively few NIINs are in all of the files, because the files were created with low-demand cut-offs in accordance with NAVSUP WSS's business rules.

Also provided are historical wholesale data files (Ellis 2017). These files have the majority of the data elements needed to run in WIOM, but they do not include the budget category, which is necessary to classify a NIIN as a particular type of material. They also contain more NIINs than the provided candidates files. Additionally, this data source is not available for the last two quarters of the period: April and July 2017. Since data is not available for the second half of FY17, the period of interest is shortened by six months, and is now Oct 2012 through June 2017.

We reorganize the provided data sets to create what we need to run WIOM and conduct our experiments in CORS: candidates files with a consistent set of NIINs for the whole test period. Since the provided candidates files have such a small set of NIINs that are present throughout, we do not use them as the basis for our new candidates files. Instead, we start with the wholesale data files. The budget category, which identifies the class of material, is still missing. In order to identify the NIINs of interest (consumable non-nuclear maritime material), we look to the requisition data and the provided candidates files. The requisition data is modified to cut out all requisitions for material that is not consumable non-nuclear maritime. The list of NIINs present in the modified requisition data now represents the list of NIINs of interest for our new candidates files. The newly created candidates files are cross referenced with this list and NIINs not in the list are deleted from the candidates files. As an additional safeguard, the NIINs present in the new candidates files are cross-referenced with the provided candidates files: any NIIN that is

identified as another class of material in any provided candidates file is deleted. The newly-formed candidates files are then cross-referenced with each other. NIINs that appear in the candidates file for each period are retained and the remainder are deleted. Lastly, the requisition data is scrubbed in the same way, and requisitions for NIINs not in the candidates files are deleted. The final result is a set of quarterly candidates files with 3,808 consumable, non-nuclear, maritime NIINs and requisition data with 106,565 requisitions.

B. METAMODEL

As input, CORS requires requisition data and WIOM outputs for each quarter of the time period being tested. To obtain the necessary WIOM outputs, we start by running WIOM for the first quarter in the time period. This run uses the candidates file for the first time period developed above, the budget figure, and the persistence parameter we are exploring. The second WIOM run for the next sequential quarter requires all the same input data plus the first WIOM solution, as it uses this information to enforce persistence. The third WIOM run requires the second WIOM solution, the fourth WIOM run requires the third WIOM solution, etc. After repeating the process for all available quarters we have a library of WIOM output. This WIOM output contains both the optimal reorder points (ROPs) and the NIIN characteristics CORS requires; namely, each NIIN's lead time (LT) and order quantity (Q). This library of 18 WIOM outputs is fed into CORS, along with the requisition data. CORS then performs its simulation and outputs system performance in terms of fill rate. Figure 2 illustrates the process.

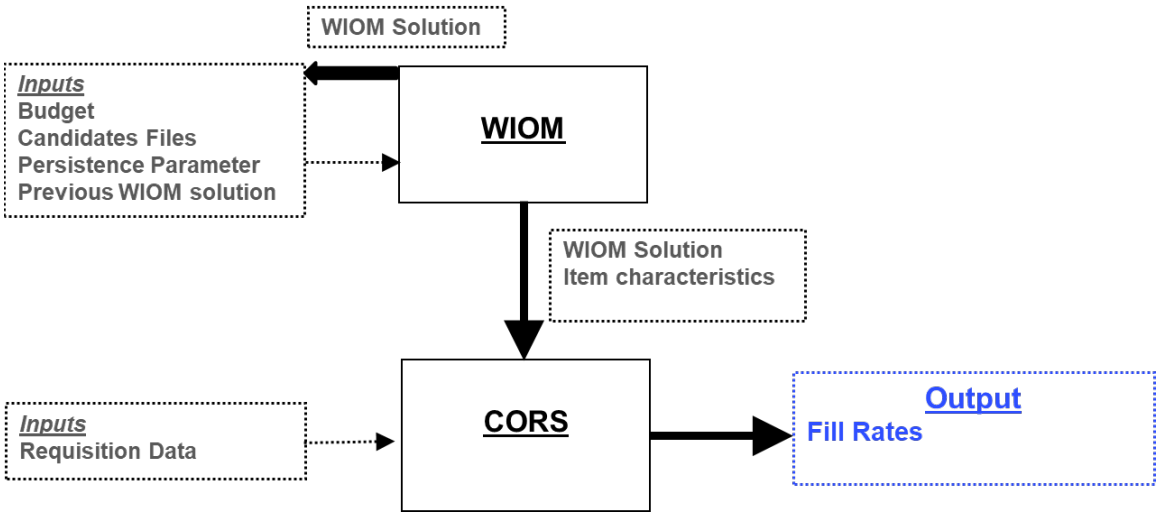

Figure 2. Metamodel Relationships

C. SIMULATION MODEL DEVELOPMENT

Using the available requisition data as input we develop CORS to model performance of the system under varying inputs of WIOM employment. As discussed, CORS works as a discrete event simulation, progressing forward in time from one event to the next. Simulating one NIIN at a time, CORS maintains an event queue with events aligned in time to trigger demand arrivals, order arrivals, and parameter changes due to new WIOM input. Each event triggers a particular logic sequence that examines the current state of the system and makes appropriate changes to the system and event queue. Table 1 shows system characteristics the simulation tracks as it runs.

Table 1. Tracked System Characteristics

	Variable	Abbreviation
1	Order Quantity	Q
2	Reorder Point	ROP
3	Lead Time	LT
4	Quantity On-Hand	Q_O/H
5	Inventory Position	IP
6	Time	t

When the simulation run begins, the Initialization Event starts. Figure 3 is a flow chart summarizing the event. First, the event adds a Parameter Reset Event to the event queue for the start of each quarter. These reset events hold information for updating Q, ROP, and LT based on the WIOM output for that period. Next, the event populates the queue with Demand Arrival Events, adding each requisition for the current NIIN onto the event queue. The Demand Events note both the date of the demand arrival and the quantity demanded for that requisition. With the event queue populated with all input data to the simulation, the event queue is sorted by date.

Figure 3. Initialization Event Flow Chart

Next, the initialization event sets the initial system conditions, assigning an initial value to each state variable. Initial Q, ROP, and LT values are assigned based on the first WIOM output in the event queue. However, we also have to assign an initial Q_O/H, IP, and trigger any order arrivals the initial IP would have caused prior to the simulation window beginning. This is a problem because the simulation runs with no warm-up period, so we must find a way to assign a starting condition that would be reasonable to find in the middle of a steady state condition. To address this issue, we make a simplifying assumption and choose to treat the inventory system as a classic inventory system. Recall that in the classical inventory model, the quantity of material on hand at any given time is distributed uniformly from zero to Q. We therefore assign the initial quantity on hand in CORS to be a uniform integer random variable (RV) between 0 and Q. This assignment ignores the possibility of material in a backorder status and the possibility of presence of stock in greater quantity than Q. But, it provides a quick way of calculating a starting condition that is on the right order of magnitude with no warm-up period. With a Q_O/H assigned, we use the same inventory model again to assign a reorder if necessary and insert it into the event queue at the appropriate time. In this model, a reorder is triggered when Q_O/H reaches ROP, and arrives precisely when Q_O/H reaches zero. We mimic this by first checking the newly assigned Q_O/H against the ROP. If Q_O/H is greater than ROP, no further action is required and the initialization event is complete. But, if Q_O/H is less than or equal to ROP, a reorder event is triggered. An order arrival event is added to the event queue with a quantity of Q. Now the question is when to have that order arrival event inserted into the event queue. If the Q_O/H is close to ROP, most of the lead time should still be left because the event would have been triggered recently. However, if Q_O/H is much lower than ROP, the order arrival would have been triggered further in the past. We therefore use the ratio of Q_O/H to ROP to calculate how much of the lead time is left and assign the date for the order arrival event. If triggered in initialization, this order arrival event is scheduled according to the following equation and IP is adjusted accordingly:

$$event\_time = start\_time + \frac{Q\_O/H}{ROP} LT$$

After the initialization event creates the event queue and sets the system state, the simulation advances from one event to the next in the event queue and completes the appropriate logic according to event type. The simplest of these event types is a parameter Reset Event. Figure 4 shows a flow chart illustrating the Reset Event actions. This event in the queue has a date as well as values for ROP, Q, and LT. This event simply reassigns the parameters ROP, Q, and LT to the appropriate values in the Reset Event. These parameters are constants that stay in effect until the next Reset Event.

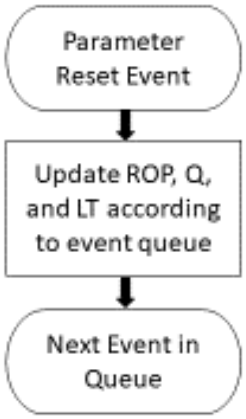

Figure 4. Parameter Reset Event Flow Chart

The next possible event type is a Demand Event. Flow chart for Demand Event logic is illustrated in Figure 5. Demands in the event queue have a date when they occur and a quantity demanded. The demand arrival event first checks the demand quantity against the Q_O/H. If the demand quantity is greater than the Q_O/H, this Demand Event is marked as being backordered. If the demand quantity is less than or equal to the Q_O/H, the event is marked as being filled. In either case, Q_O/H is then decremented by the demand quantity (negative Q_O/H representing items in a backorder status). IP is also decremented by demand quantity. The logic then checks IP against ROP. If IP is less than or equal to ROP, an Order Arrival Event is scheduled to occur in one LT, and the Order Arrival Event is scheduled with a quantity of Q. IP is increased by Q. The logic rechecks IP against ROP and continues these steps until IP is greater than ROP. At this point the event is complete and the next event in the queue is processed.

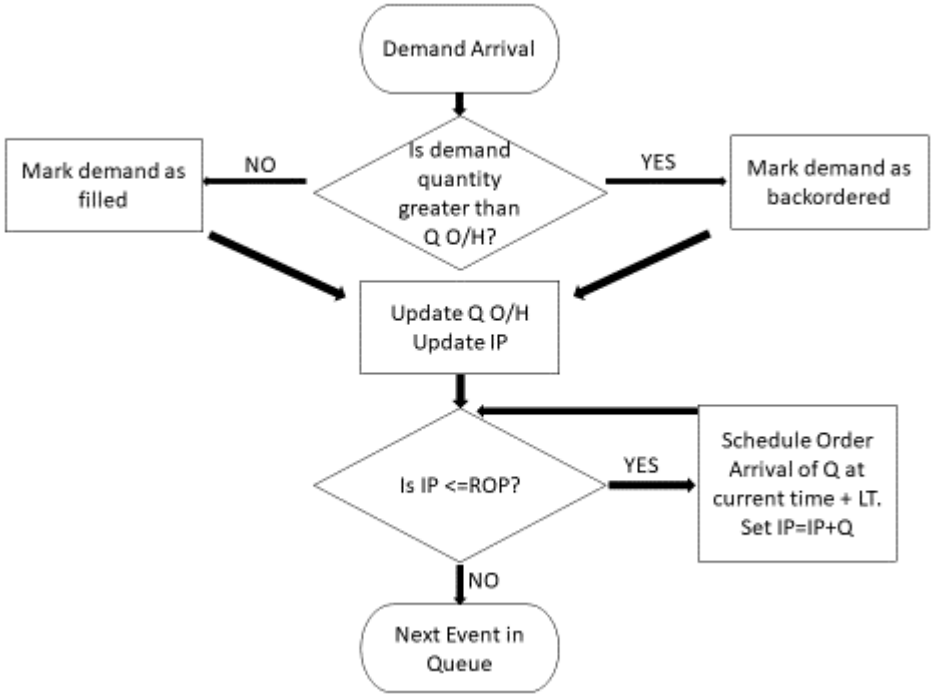

Figure 5. Demand Arrival Event Flow Chart

The final possible event type is the Order Arrival Event, illustrated in Figure 6. The Order Arrival event checks whether the Q_O/H is positive, negative, or zero. If it is negative, the logic runs a process to clear existing backorders as feasible with the quantity of the order arrival. If Q_O/H is zero or greater, this process is skipped. Either way, the logic then updates the Q_O/H, adding the order quantity Q to Q_O/H.

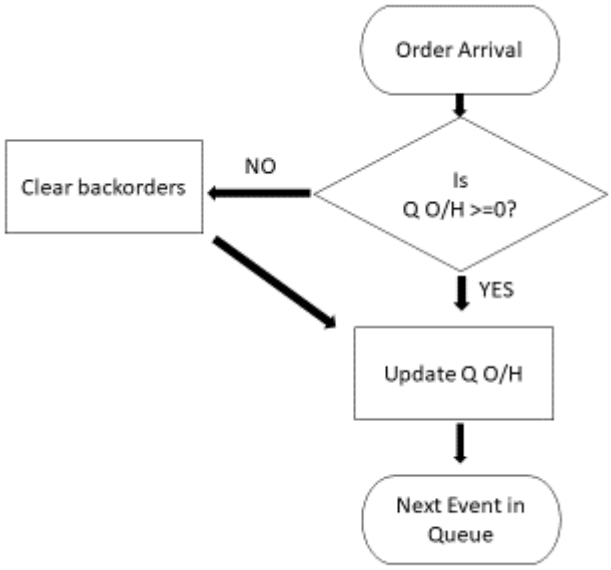

Figure 6. Order Arrival Event Flow Chart

We implement the simulation logic in the R programming language (R Core Team 2016) to run CORS. Additional logic not detailed here is included to record statistics of system performance.

D. MODEL OUTPUT

The model outputs information that can be used to calculate fill rate in a variety of ways. First, the model outputs the overall fill rate for each NIIN for the entire simulation. Next the model outputs aggregate data for all NIINs that can be used to calculate the fill rates for a number of time frames. For each month, the total number of requisitions filled (across all NIINs) and the total number of requisitions received are both recorded. With these pieces of data, aggregate demand weighted fill rates can be calculated for any periodicity that is a multiple of months (i.e., quarterly, annually, etc.). Finally, the model outputs the average length all backordered requisitions stayed in a backorder status.

THIS PAGE INTENTIONALLY LEFT BLANK

III. ANALYSIS

A. OPERATING CONCEPTS EXPLORED

With a working CORS, we next must decide what concepts of operations to simulate. We have two items we wish to explore: run periodicity and the persistence parameter. Based on our assumptions discussed in Chapter I, we only consider periodicities of quarterly, semi-annually, and annually. For the persistence parameter, we choose to use parameters that roughly correlate to none, low, medium, and high. The low, medium, and high values of persistence are 0.1, 1.0, and 5.0, respectively. We chose these numbers based on our observation of the impact of persistence parameter on WIOM's predicted fill rate. Using these 3 periodicities and 4 persistence parameters, there are 12 total possible combinations, all of which we include in our experiments. Additionally, we explore the possibility of a hybrid approach, where WIOM is run every quarter, but with different persistence parameters. In this hybrid idea, persistence is turned off in one model run per year in order for the solution to "reset" and adapt to any drift that has occurred in the demand distributions. The other three quarters the persistence parameter is set at the low, medium, or high level. These three hybrid designs bring the total experiment to 15 concepts. Table 2 shows the list of settings for the 15 designs and the resulting overall fill rates achieved by each, as simulated in CORS. Note that WIOM does not directly maximize fill rate; rather, it minimizes a series of piecewise linear penalties associated with negative deviations from fill rate goals. Nonetheless, overall fill rate provides a simple aggregate figure of merit by which to judge system performance.

Table 2. Overall Simulated Fill Rates

Design	Periodicity	Persistence	Overall Fill Rate
1	Annual	0.0	51.77%
2	Annual	0.1	51.88%
3	Annual	1.0	51.85%
4	Annual	5.0	51.74%
5	Semi-annual	0.0	58.34%
6	Semi-annual	0.1	58.08%
7	Semi-annual	1.0	58.45%
8	Semi-annual	5.0	58.01%
9	Quarterly	0.0	61.57%
10	Quarterly	0.1	61.16%
11	Quarterly	1.0	61.43%
12	Quarterly	5.0	60.90%
13	Annual/Quarterly	0.0/0.1	61.53%
14	Annual/Quarterly	0.0/1.0	61.46%
15	Annual/Quarterly	0.0/5.0	61.32%

B. TIME TO DIVERGE

The results of our experiment shown in Table 2 indicate a clear delineation between certain concepts of operation in the overall fill rates across the simulation. The greatest difference occurs for designs 4 and 9, which differ by 9.83%. However, these concepts of operation, the best and the worst performing in the simulation, do not show any immediate difference in fill rates during the early parts of the simulation. Figure 7 shows the difference between monthly fill rates for these two designs, calculated as the monthly fill rate for design 9 minus the fill rate for design 4.

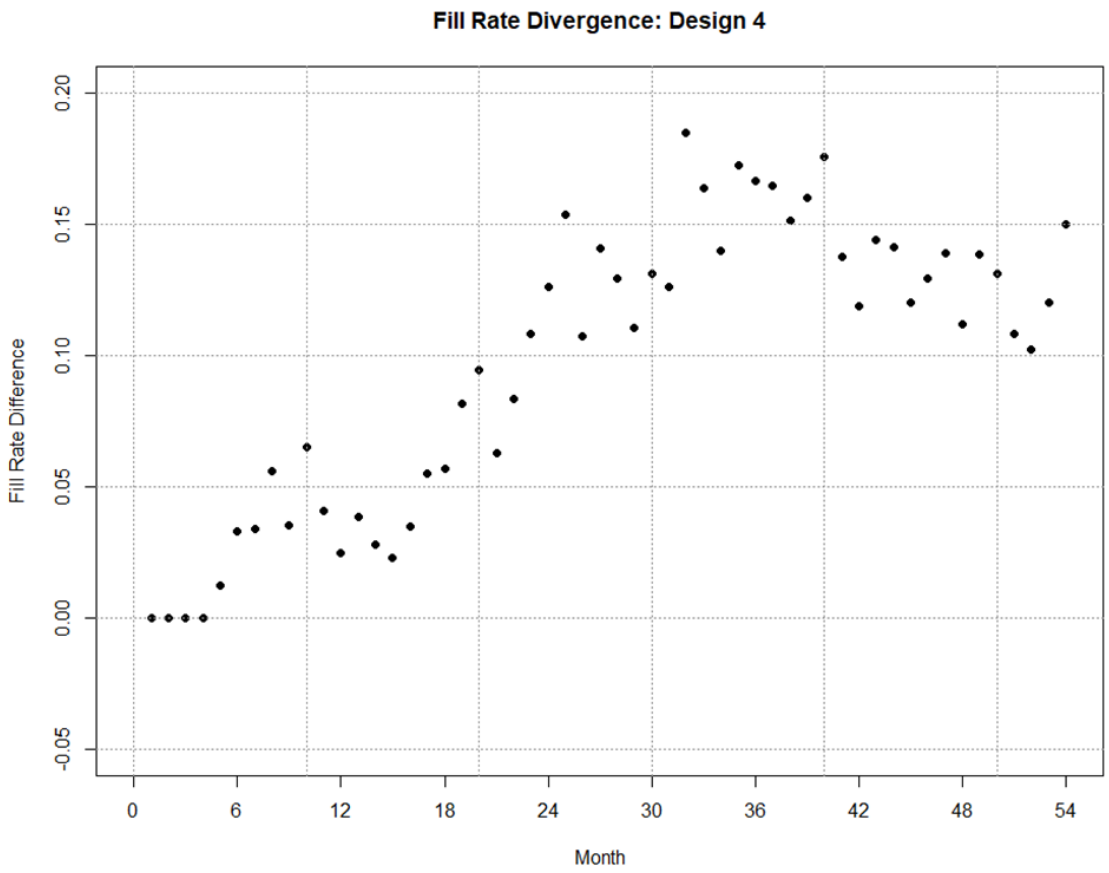

Figure 7. Monthly Difference in Simulated Fill Rate between Designs 4 and 9

We do not observe a difference of about 10% (roughly the overall difference between the two designs) until month 20. It takes six months for the designs to start to diverge and almost two years until we gain an idea of the performance differences between these two designs. The time to show clear divergence is even longer with a design that has less degradation from the best.

Figure 8 shows the difference in monthly fill rates between designs 8 and 9, which have an overall fill rate difference of 3.56%. Here the first deviations are at month 6, 7, and 8, but the difference is less in subsequent months. Divergence is not clear until about month 18.

The key insight here is that the system takes a long time to show differences in performance. Based on what we see here, we expect at least two quarters before any impact of a WIOM implementation is felt, and much longer before the degree of impact is shown.

This makes intuitive sense as well, as the average lead time across the NIINs tested is a little more than a year.

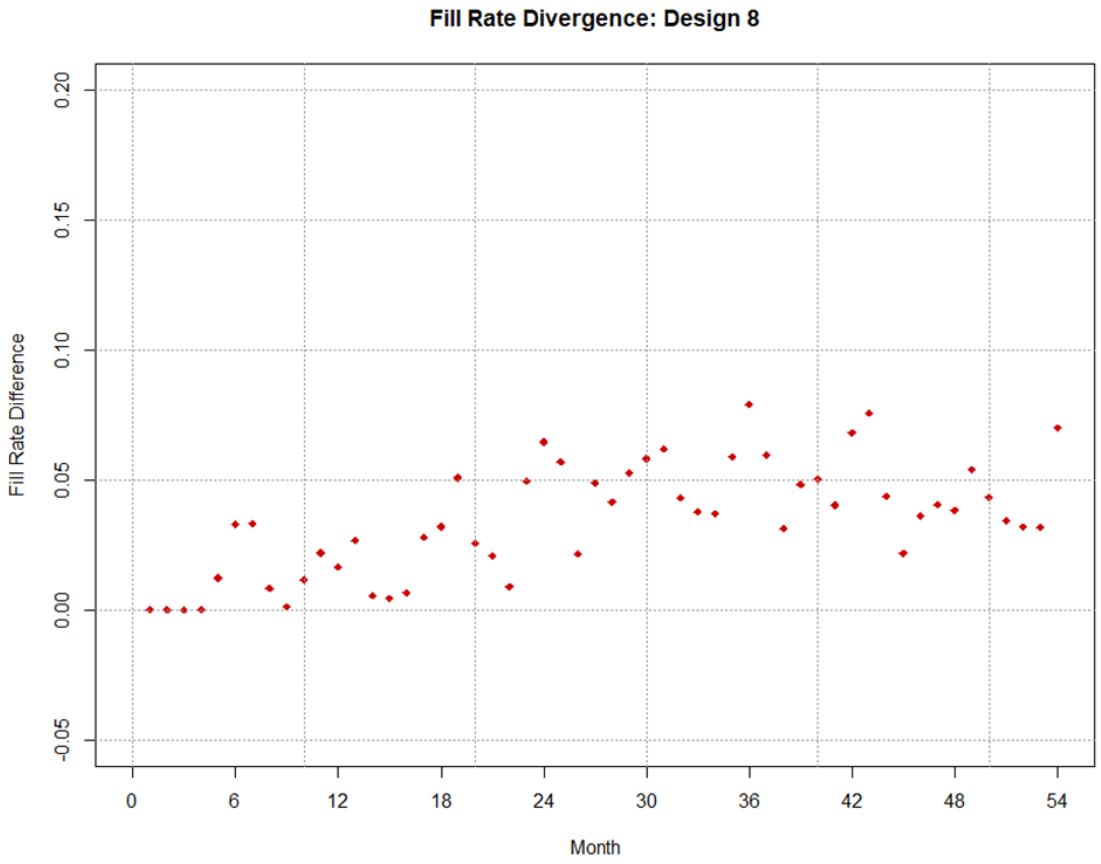

Figure 8. Monthly Difference in Simulated Fill Rate between Designs 8 and 9

C. EFFECT OF PERIODICITY

One of the goals of this thesis is to test whether running WIOM at different periodicities affects system performance. Our results indicate a clear degradation in system performance with longer periodicities. At any level of persistence, performance degrades with increases in time between WIOM runs. Table 3 shows overall fill rates of quarterly, semi-annual, and annual periodicities with the persistence parameter set to zero.

Table 3. Fill Rates by Periodicity

Design	Periodicity	Overall Fill Rate	Degradation from Best
1	Annual	51.77%	9.80%
5	Semi-Annual	58.34%	3.23%
9	Quarterly	61.57%	0.00%

Differences between periodicities are similar at all tested levels of persistence. We see a clear degradation in fill rate from a quarterly concept to a semi-annual concept, and a dramatic degradation from quarterly to annual. This degradation with longer periodicities provides evidence of the system changing over time. This change over time seems to give any WIOM solution an inherent "shelf-life." Operating the system with an overaged WIOM solution delivers sub-optimal performance.

D. EFFECT OF PERSISTENCE PARAMETER

The next goal of the thesis is to quantify the trade-off between churn and fill rate performance. Achieving this goal involves a two-step process. The change in input to the model to vary churn is the persistence parameter. However, the persistence parameter does not directly set a certain level of churn. Rather, it is a change in the weighting of the objective function for the WIOM optimization model. So, we must first analyze the effect of the persistence parameter on churn, and then analyze the effect on fill rate performance. It is important to note here that we are comparing churn, which is calculated in WIOM, against simulated fill rate performance, which is not. The purpose here is not to compare the relative values of the two terms in WIOM's objective function. Rather, our goal in this study is to compare churn against simulated system performance. Having shown that annual and semi-annual concepts perform poorly, we restrict the persistence analysis to quarterly periodicities only.

1. Effect of Persistence Parameter on Churn

The persistence parameter in WIOM enforces persistence by applying a penalty when the safety stock of a NIIN differs from the previous safety stock level. The safety stock is the expected quantity on hand when a reorder arrives. The penalty for any given

NIIN can be defined by the following expression, where $\hat{s}^0$ is the NIIN's safety stock in the incumbent solution and s is the safety stock in the new solution:

$$\frac{\left|\hat{s}^0 - s\right|}{\hat{s}^0 + 1}$$

This expression calculates a penalty that is proportional to the relative magnitude of the change. For example, a change of solution from 9 to 10 incurs a penalty of 0.1, while a change from 9 to 19 incurs a penalty of 1.0. A NIIN with no solution change incurs no penalty. The penalties from all NIINs are summed in the objective function. If we define the set of NIINs as I and index them as $i \in I$, we can express the summation of the penalties by the following expression:

$$\sum_{i \in I} \frac{\left|\hat{s}^0 - s\right|}{\hat{s}^0 + 1}$$

This expression can be used to define the total churn present in a given solution. We can then compare values from different solutions. If one solution has a lower value of this expression, it represents less churn (an improvement). WIOM uses a mathematically equivalent, but different, expression to define churn. The expression presented here is used instead of WIOM's for simplicity. WIOM's expression avoids using an absolute value in order to make the optimization problem linear, but requires multiple constraints in order to do so.

To compare the churn across our quarterly designs, we compute this value for every quarter, and take the mean value across the simulation time period for each concept of operation design. The results of these calculations are shown in Table 4.

Table 4. Churn Value by Design

Design	Persistence	Average Churn
9	0.0	5,673
10	0.1	600
11	1.0	154
12	5.0	50
13	0.0/0.1	2,512
14	0.0/1.0	2,328
15	0.0/5.0	2,137

The designs using a constant persistence parameter every quarter show a clear reduction in churn with increasing persistence parameter. The highest persistence parameter tested has, on average, less than 1% the churn present with the parameter set to 0.0. The impact of the parameter is less obvious on the hybrid concept designs: 13, 14, and 15. In these designs, the parameter is set to 0.0 once a year, and the other three quarters it is set as indicated in Table 4. Here the average churn decreases marginally from one design to the next, and each hybrid design has more average churn than all other designs except design 9, which uses a persistence parameter of 0.0 throughout. Looking more closely at the hybrid designs, we see that they have very high churn rates the one time of year that they use a parameter of 0.0. Table 5 shows the average churn rates of these designs when the parameter is equal to zero and when it is not.

Table 5. Churn Values for Hybrid Designs

Design	Persistence	Average with Zero Persistence	Average with Positive Persistence	Overall
13	0.0/0.1	8,816	572	2,512
14	0.0/1.0	9,391	155	2,328
15	0.0/5.0	8,925	49	2,137

Looking at Table 5, we make two observations. First, in the quarters when persistence above 0.0 is used, average churn for designs 13, 14, and 15 is very similar to average churn for designs 10, 11, and 12, respectively (see Table 4). The next observation is that the large overall average churn for the hybrid designs comes from the annual runs

with persistence set to 0.0. In these designs churn is very high during the annual "reset" of the WIOM solution but effectively reduced during other quarters.

The above analysis shows that the persistence parameter reduces churn. However, this definition of churn is abstract and mathematical, and there is no immediate understanding of what its values mean to the system. An alternate way to express churn that is more intuitive is to define it as the proportion of NIINs that had any change in safety stock. While WIOM does not use this definition (nor does it pursue such a goal in the objective function), we expect this measurement to decrease in concert with WIOM's definition of churn, and we wish to know if it does not. Using this alternate definition of churn as a proportion, we calculate the average across the simulation period for the different designs in Table 6. As expected, increasing the persistence parameter reduces the proportion of NIINs that have a change in safety stock. However, the reduction is less dramatic than that reflected in the churn formula. The churn formula calculated churn at persistence parameter level 5.0 as less than 1% of the churn at persistence parameter 0.0. Using this alternate definition, the reduced churn for the same designs is about 25%.

Table 6. Churn as Percentage of Items with Change

Design	Persistence	Average Items with Churn
9	0.0	39.99%
10	0.1	30.72%
11	1.0	16.12%
12	5.0	10.35%
13	0.0/0.1	36.73%
14	0.0/1.0	27.34%
15	0.0/5.0	23.10%

As in Table 5, we also calculate the rates by phase for the hybrid designs. These results are presented in Table 7. Results using the new definition of churn are much like when using the original definition. Churn is effectively reduced when using the parameter and large amounts of churn are seen at the annual "reset" when the persistence parameter is set to 0.0.

Table 7. Churn as Percentage of Items with Change for Hybrid Designs

Design	Persistence	Average with Zero Persistence	Average with Positive Persistence	Overall
13	0.0/0.1	55.00%	31.10%	36.73%
14	0.0/1.0	62.11%	16.65%	27.34%
15	0.0/5.0	64.49%	10.36%	23.10%

A third way to define churn is by dollar value. For any given NIIN, we can define a change in the stock cost as the absolute value of the change in the solution times the unit cost of that NIIN. This dollar value can be an effective way to think of the difference between one solution and another. However, as before, this is not the way WIOM pursues churn reduction. Using this definition, we create Tables 8 and 9, equivalent to Tables 6 and 7 but using the dollar value definition of churn. We see similar behavior to results seen using the other two definitions.

Table 8. Churn as Dollar Value

Design	Persistence	Average Churn (Millions $)
9	0.0	6.29
10	0.1	4.68
11	1.0	2.83
12	5.0	1.95
13	0.0/0.1	5.49
14	0.0/1.0	4.55
15	0.0/5.0	4.08

Table 9. Churn as Dollar Value for Hybrid Designs

Design	Persistence	Average with Zero Persistence	Average with Positive Persistence	Overall
13	0.0/0.1	7.05	5.01	5.49
14	0.0/1.0	8.69	3.28	4.55
15	0.0/5.0	10.18	2.21	4.08

2. Churn versus Fill Rate Trade-off

Having calculated persistence, we can now address one of the thesis's fundamental questions: what is the trade-off between churn and fill rate performance? For this analysis we use WIOM's calculation of churn. We start by looking at the relationship between churn value and fill rate for our seven quarterly designs. A graph of these points is presented in Figure 9. However, it is important to note that we are graphing the simulated fill rates achieved over the time period. We *are not* attempting to find the Pareto curve of efficient solutions, which would be applicable to the two components of the objective value calculated by WIOM. Rather, we are trying to get an idea of the trade-off of between fill rate performance and churn achieved in a production-type environment.

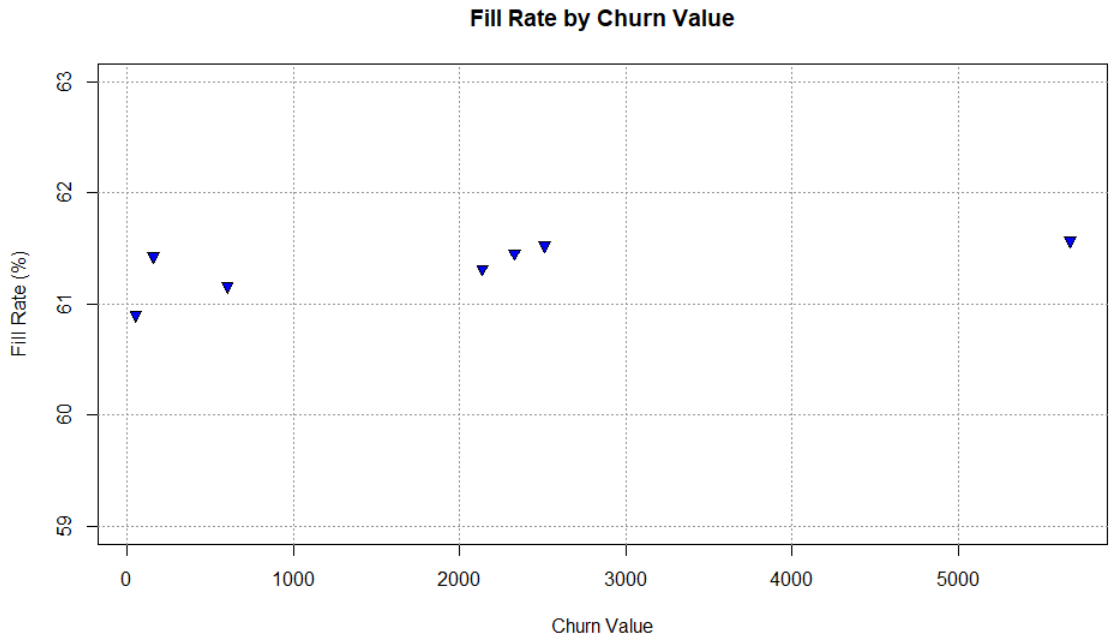

Figure 9. Graph of Fill Rate by Churn Value

It appears that there is a very slight increase (improvement) in fill rate associated with an increase (degradation) in churn, which is what we expect. But, we have few data points and the increase is very slight. Reductions (improvement) in churn are very "cheap" in terms of fill rate for these levels of persistence parameter for this set of historical demand.

E. PERSISTENCE PARAMETER FURTHER EXPLORATION

Based on the results of the quarterly concepts from our original experimental design, we observe only a small trade-off relationship between churn value and simulated fill rate. However, we know that at some level a larger trade-off exists. The annual and semi-annual designs effectively have churn-free solutions in the quarters that WIOM is not run. These designs have clear degradation in fill rate compared to the quarterly designs. Therefore, there must be some threshold of churn improvement that causes greater levels of simulated fill rate degradation. However, the persistence parameters we explored did not create churn reduction that crossed that threshold. We therefore conduct a new experiment with higher settings of the persistence parameter to find this threshold and find a steeper trade-off between churn and fill rate.

1. New Concept Testing

We add three new concepts of operation to our experiment. We use quarterly runs with the persistence parameter set at 10, 100, and 1000. For this analysis we exclude the hybrid designs. Our new design is presented in Table 10. Using these designs we perform WIOM runs as applicable and run the output in CORS to conduct the experiment.

Table 10. Follow-on Concepts Testing (All Quarterly)

Design	Persistence
1B	0.0
2B	0.1
3B	1.0
4B	5.0
5B	10.0
6B	100.0
7B	1,000.0

2. Effect on Churn

Despite the large increases in the persistence parameter for designs 5B, 6B, and 7B, there is relatively little effect on churn as measured by any of our three definitions. Churn values are presented in Table 11.

Table 11. Churn for Follow-on Concepts Testing

Design	Persistence	Average Churn	Average Items with Churn	Average Churn (Millions $)
1B	0.0	5,673	39.99%	6.29
2B	0.1	600	30.72%	4.68
3B	1.0	154	16.12%	2.83
4B	5.0	50	10.35%	1.95
5B	10.0	33	8.69%	1.76
6B	100.0	29	8.09%	1.71
7B	1,000.0	29	8.08%	1.71

It appears that increasing the persistence parameter above 5.0 only marginally decreases churn, and increasing it over 10.0 affects churn only modestly. We observe severe decreasing marginal returns for increasing the persistence parameter. Graphing average churn against the persistence parameter for designs 1B-5B in Figure 11 shows this phenomenon clearly. There is an obvious “knee” in the curve.

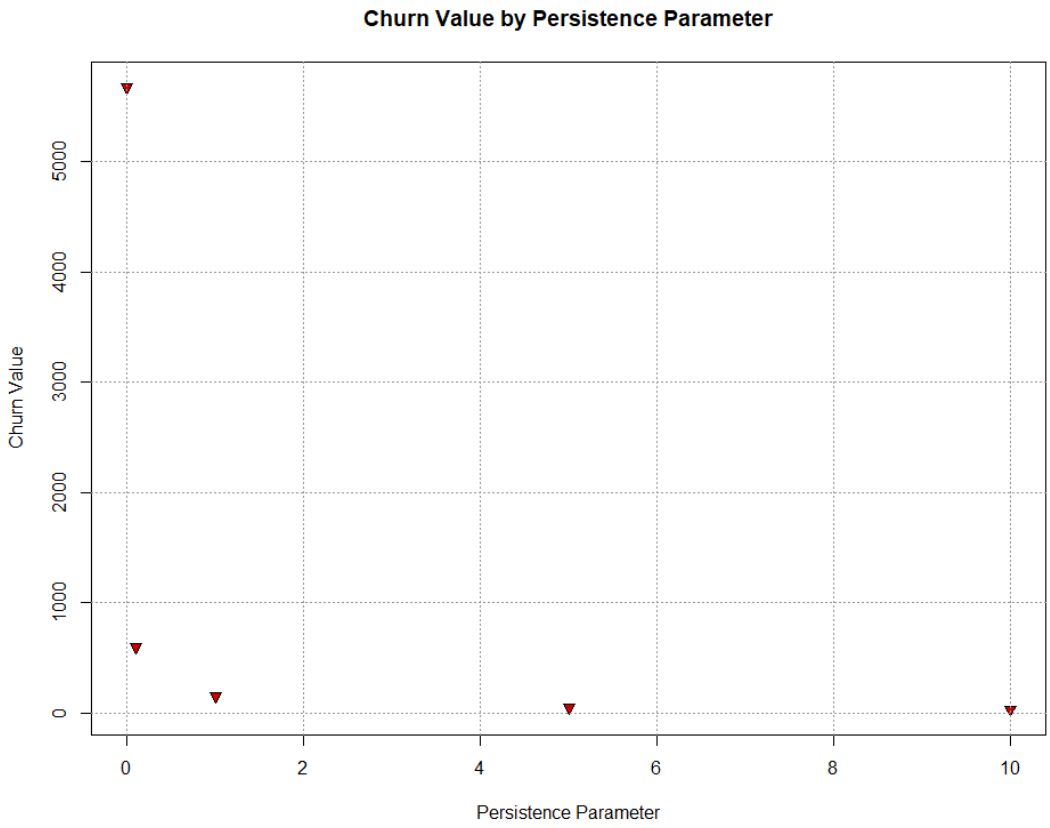

Figure 10. Graph of Churn Value by Persistence Parameter

3. Effect on Fill Rate

As the increase in persistence parameter has little effect on churn, it also has little effect on fill rate performance. Fill rate performance by persistence parameter is shown in Table 12. Only marginal decreases in fill rate are observed.

Table 12. Follow-on Testing Fill Rate Results

Design	Persistence	Overall Fill rate
1B	0.0	61.57%
2B	0.1	61.16%
3B	1.0	61.43%
4B	5.0	60.90%
5B	10.0	60.50%
6B	100.0	60.29%
7B	1,000.0	60.26%

THIS PAGE INTENTIONALLY LEFT BLANK

IV. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

This thesis develops a new simulation model, CORS, in order to explore the effects of different concepts of operation for WIOM implementation. These concepts of operation vary in terms of the periodicity that WIOM is executed and the persistence parameter used. We explore a variety of different concepts of operations using CORS and we measure system performance for each design in terms of simulated fill rate and churn. Through the course of this research we have gained several key insights into NAVSUP WSS's wholesale inventory system.

The first insight we gain is that it takes time for different implementation concepts to differentiate in terms of fill rate. Even very clearly different solutions take at least six months to produce different fill rates. It takes even longer for the magnitude of the difference to become clear. This insight is important because it reminds us to be cautious in judging the performance of the system in the short term.

Our next key insight into the system is that WIOM solutions have a short shelf life. The system changes sufficiently over time that there are clear degradation to fill rate performance for semi-annual designs and dramatic degradation for annual designs. While different solutions take time to diverge, it is important for the optimization model to be able to adjust to changes in the underlying demand structure quickly. We see no reason to recommend a change to the quarterly periodicity that NAVSUP WSS currently uses.

Perhaps our most important finding is that, for the historical demand considered, churn can be drastically reduced without sacrificing system performance in terms of fill rate. By implementing the use of the persistence parameter, NAVSUP WSS can gain significant improvement in churn, which reduces administrative burden in contracting and improves explainability of WIOM results to senior leadership. All this improvement can be gained without sacrificing fill rate performance and support to the fleet.

Our final finding is unexpected. It appears that WIOM has a limit to how far it can enforce persistence. Beyond a certain point, increasing the persistence parameter has no

practical effect on churn. Even increasing the persistence parameter several orders of magnitude has virtually no effect on churn. This may be due to a WIOM solution in one quarter not being feasible in a following quarter. For instance, this could occur due to WIOM's budget constraint. If the incumbent solution is too costly for the current budget, the lowest feasible value of churn will be strictly positive. Or, this phenomenon may be due to optimality tolerance. This study used a relative optimality gap of 3% when solving WIOM. More testing is required for a definitive conclusion.

While we noticed several important features in the system, it is also important to be clear about what we did not find. Our first important caveat concerns the lack of reduction in fill rate with increases in the persistence parameter. In this particular case, we observed that the limit that persistence could be enforced was above the critical threshold where it would impact simulated fill rates. In this way, we could increase the persistence parameter to an arbitrarily large number and not affect fill rates. However, we do not have evidence that this is true generally. It may well be that this is simply a happy coincidence of this particular type of material, for these demands, and at this budget level.

The next important caveat is that our conclusions are based on only 4.5 years of data. We showed that simulated fill rates did not degrade with increases in the persistence parameter *for this time period only*. We also showed that the difference between a good and bad concept of operations takes time to develop. It is possible that some level of persistence does impact long-term fill rates when viewed from a longer term horizon.

B. FOLLOW-ON RESEARCH RECOMMENDATIONS

This thesis explored NAVSUP WSS's wholesale inventory system in several ways. However, there is much more to be done. We present the following as recommended areas for follow on study and research.

First, the CORS model was only applied to maritime, non-nuclear, consumable material. Without change to the simulation, CORS can be used to do testing on other consumable material types, namely aviation material and maritime nuclear material. These datasets are considerably different in terms of demand, budget, and cost. Additionally, it is

possible to revise the simulation to accommodate repairable material and perform tests on both maritime and aviation repairable material.

We also recommend revising the model to include more elements of the inventory management system. Two key elements of the wholesale inventory system not modeled in CORS are substitute NIINs and demand priority. CORS does not fill requisitions with any NIIN but the one requisitioned, while the actual system can fill requisitions with alternate or substitute NIINs if they are available. CORS does not use any demand priority scheme, and instead treats each requisition as equal. Follow on research including these elements into the model will give greater granularity to system performance.

Using CORS we are limited to deterministic historical demand. This restricts how much we can test the robustness of the system to changes in demand and limits us in terms of time horizon we can test. We recommend future research find a way to revise the metamodel to make demand arrivals stochastic and to run the simulation for a longer period of time.

We also recommend future research in revising WIOM. We calculated churn by proportion of NIINs that were unchanged and by dollar value change in this thesis. The optimization model can be amended to reduce churn according to one of these (or a different) definitions.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX. CORS CODE

```
rm(list = ls())
setwd("C:/Users/Sean/Desktop/Thesis/Testing Environment")
require(lubridate)
require(plyr)
set.seed(736)
#FY13 Req Data
req13=read.csv("FY13 Fill Rate Data.csv") #raw data is WSS provided excel files resaved
as .csv files
req13=req13[req13$CR=="Consumables",]
req13=req13[req13$SOURCE=="Maritime",]
req13=req13[req13$NUC=="Non-Nuclear",]
#subset out columns not of interest
req13=req13[,c(8,21,69,27)]
req13$historical=0
req13[(req13$HIT.MISS.FINAL=="H"),]$historical=1
req13=req13[,c(1,2,3,5)]
req13$JUL.DATE=as.integer(as.character(req13$JUL.DATE))
#FY14 Req data
req14=read.csv("FY14 Fill Rate Data.csv")
req14=req14[req14$CR=="Consumables",]
req14=req14[req14$SOURCE=="Maritime",]
req14=req14[req14$NUC=="Non-Nuclear",]
#subset out columns not of interest
req14=req14[,c(8,21,69,27)]
req14$historical=0
req14[(req14$HIT.MISS.FINAL=="H"),]$historical=1
req14=req14[,c(1,2,3,5)]
req14$JUL.DATE=as.integer(as.character(req14$JUL.DATE))
#FY15 Req data
req15=read.csv("FY15 Fill Rate Data.csv")
req15[76:91]=list(NULL) #remove excess columns
#subset requisition data into Maritime Consumables only
req15=req15[req15$CR=="Consumables",]
req15=req15[req15$SOURCE=="Maritime",]
req15=req15[req15$NUC=="Non-Nuclear",]
#subset out columns not of interest
req15=req15[,c(8,21,69,27)]
req15$historical=0
req15[(req15$HIT.MISS.FINAL=="H"),]$historical=1
req15=req15[,c(1,2,3,5)]
req15$JUL.DATE=as.integer(as.character(req15$JUL.DATE))
```

```
#FY16 req data
req16=read.csv("FY16 Fill Rate Data.csv")
#subset requisition data into Maritime Consumables only
req16=req16[req16$CR=="Consumables",]
req16=req16[req16$SOURCE=="Maritime",]
req16=req16[req16$NUC=="Non-Nuclear",]
#subset out columns not of interest
req16=req16[,c(8,21,69,27)]
req16$historical=0
req16[(req16$HIT.MISS.FINAL=="H"),]$historical=1
req16=req16[,c(1,2,3,5)]
req16$JUL.DATE=as.integer(as.character(req16$JUL.DATE))
#FY17 Req Data
req17=read.csv("FY17 Fill Rate Data.csv")
req17=req17[req17$CR=="Consumables",]
req17=req17[req17$SOURCE=="Maritime",]
req17=req17[req17$NUC=="Non-Nuclear",]
#subset out columns not of interest
req17=req17[,c(8,21,69,27)]
req17$historical=0
req17[(req17$HIT.MISS.FINAL=="H"),]$historical=1
req17=req17[,c(1,2,3,5)]
req17$JUL.DATE=as.integer(as.character(req17$JUL.DATE))
####bind multiple req datas together here####
reqs=rbind(req13,req14,req15,req16,req17)
sum(is.na(reqs$JUL.DATE)) #see how many NAs are created when cleaning data
#delete unused dfs to save memory
req13=NULL
req14=NULL
req15=NULL
req16=NULL
req17=NULL
####add new WIOM output files here to add data####
#WIOM runs
WIOM_1210=read.csv("WIOM_1210.csv")
WIOM_1210=WIOM_1210[,c(1,4,3,32)]
WIOM_1301=read.csv("WIOM_1301.csv")
WIOM_1301=WIOM_1301[,c(1,4,3,32)]
WIOM_1304=read.csv("WIOM_1304.csv")
WIOM_1304=WIOM_1304[,c(1,4,3,32)]
WIOM_1307=read.csv("WIOM_1307.csv")
WIOM_1307=WIOM_1307[,c(1,4,3,32)]
WIOM_1310=read.csv("WIOM_1310.csv")
WIOM_1310=WIOM_1310[,c(1,4,3,32)]
WIOM_1401=read.csv("WIOM_1401.csv")
```

```
WIOM_1401=WIOM_1401[,c(1,4,3,32)]
WIOM_1404=read.csv("WIOM_1404.csv")
WIOM_1404=WIOM_1404[,c(1,4,3,32)]
WIOM_1407=read.csv("WIOM_1407.csv")
WIOM_1407=WIOM_1407[,c(1,4,3,32)]
WIOM_1410=read.csv("WIOM_1410.csv")
WIOM_1410=WIOM_1410[,c(1,4,3,32)]
WIOM_1501=read.csv("WIOM_1501.csv")
WIOM_1501=WIOM_1501[,c(1,4,3,32)]
WIOM_1504=read.csv("WIOM_1504.csv")
WIOM_1504=WIOM_1504[,c(1,4,3,32)]
WIOM_1507=read.csv("WIOM_1507.csv")
WIOM_1507=WIOM_1507[,c(1,4,3,32)]
WIOM_1510=read.csv("WIOM_1510.csv")
WIOM_1510=WIOM_1510[,c(1,4,3,32)]
WIOM_1601=read.csv("WIOM_1601.csv")
WIOM_1601=WIOM_1601[,c(1,4,3,32)]
WIOM_1604=read.csv("WIOM_1604.csv")
WIOM_1604=WIOM_1604[,c(1,4,3,32)]
WIOM_1607=read.csv("WIOM_1607.csv")
WIOM_1607=WIOM_1607[,c(1,4,3,32)]
WIOM_1610=read.csv("WIOM_1610.csv")
WIOM_1610=WIOM_1610[,c(1,4,3,32)]
WIOM_1701=read.csv("WIOM_1701.csv")
WIOM_1701=WIOM_1701[,c(1,4,3,32)]
RECAP_total=NULL #ensure summary dataframe is empty when beginning
#order requisitions by NIIN
reqs=reqs[order(reqs$NIIN),]
row.names(reqs)=1:nrow(reqs)
#format order quantity as a numeric, removing EA or other non quantitative info
reqs$ORDER.QTY=as.character(reqs$ORDER.QTY)
reqs$ORDER.QTY=do.call(rbind,strsplit(reqs$ORDER.QTY,' '))[,1]
reqs$ORDER.QTY=as.numeric(reqs$ORDER.QTY)
#sum(is.na(reqs$ORDER.QTY)) #see how many NAs are created when cleaning order qty
data
#split julian date into year and julian date
reqs$JUL.DATE=as.character(reqs$JUL.DATE)
reqs$Year=substring(reqs$JUL.DATE, 1,1)
reqs$JUL.DATE=substring(reqs$JUL.DATE, 2,4)
reqs$Year=as.numeric(reqs$Year)+2010
#create a time stamp equal to midnight of jan 1 of the year for the row (in ZULU time)
reqs$Start=paste(as.character(reqs$Year),"-01-01 00:00:00",sep="")
#convert the character to date format using lubridates ymd_hms() function
reqs$Start=ymd_hms(reqs$Start, tz = "UTC")
#add numeric dates to convert julian date into regular date format
```

```
reqs$Date=reqs$Start+(as.numeric(reqs$JUL.DATE)-1)*24*60*60
#subset out columns that were used in formatting dates that are no longer required
reqs=reqs[,c(7,1,3,4)]
#subset out requisitions that are prior to or after our period of interest
####adjust date when adding data####
reqs=reqs[(reqs$Date>as.Date("2012-09-30")),]
reqs=reqs[(reqs$Date<as.Date("2017-04-01")),]
row.names(reqs)=1:nrow(reqs)
NIIN_list=read.csv("CAN_1210.txt")
#check to see if the NIINs in req file are also in WIOM_ files
reqs$InCanFile= (reqs$NIIN %in% NIIN_list$NIIN)
numreqs=sum(reqs$InCanFile)
#subset out the NIINs without WIOM_ file info
reqs=reqs[(reqs$InCanFile==TRUE),]
#subset out rows with missing information, format and clean dataframe
reqs=na.omit(reqs)
reqs$NIIN=factor(reqs$NIIN)
reqs$InCanFile=NULL
#remove historical info for experiment
reqs$historical=NULL
#set up summary data frame
####adjust number of months based on data being ran####
dfsum=data.frame(
  Month1=numeric(),
  Month2=numeric(),
  Month3=numeric(),
  Month4=numeric(),
  Month5=numeric(),
  Month6=numeric(),
  Month7=numeric(),
  Month8=numeric(),
  Month9=numeric(),
  Month10=numeric(),
  Month11=numeric(),
  Month12=numeric(),
  Month13=numeric(),
  Month14=numeric(),
  Month15=numeric(),
  Month16=numeric(),
  Month17=numeric(),
  Month18=numeric(),
  Month19=numeric(),
  Month20=numeric(),
  Month21=numeric(),
  Month22=numeric(),
```

```
  Month23=numeric(),
  Month24=numeric(),
  Month25=numeric(),
  Month26=numeric(),
  Month27=numeric(),
  Month28=numeric(),
  Month29=numeric(),
  Month30=numeric(),
  Month31=numeric(),
  Month32=numeric(),
  Month33=numeric(),
  Month34=numeric(),
  Month35=numeric(),
  Month36=numeric(),
  Month37=numeric(),
  Month38=numeric(),
  Month39=numeric(),
  Month40=numeric(),
  Month41=numeric(),
  Month42=numeric(),
  Month43=numeric(),
  Month44=numeric(),
  Month45=numeric(),
  Month46=numeric(),
  Month47=numeric(),
  Month48=numeric(),
  Month49=numeric(),
  Month50=numeric(),
  Month51=numeric(),
  Month52=numeric(),
  Month53=numeric(),
  Month54=numeric(),
  TotalFillRate=numeric(),
  AvgBB=numeric(),
  stringsAsFactors=FALSE)
dfactualfills=data.frame(
  Month1=numeric(),
  Month2=numeric(),
  Month3=numeric(),
  Month4=numeric(),
  Month5=numeric(),
  Month6=numeric(),
  Month7=numeric(),
  Month8=numeric(),
  Month9=numeric(),
```

Month10=numeric(),
Month11=numeric(),
Month12=numeric(),
Month13=numeric(),
Month14=numeric(),
Month15=numeric(),
Month16=numeric(),
Month17=numeric(),
Month18=numeric(),
Month19=numeric(),
Month20=numeric(),
Month21=numeric(),
Month22=numeric(),
Month23=numeric(),
Month24=numeric(),
Month25=numeric(),
Month26=numeric(),
Month27=numeric(),
Month28=numeric(),
Month29=numeric(),
Month30=numeric(),
Month31=numeric(),
Month32=numeric(),
Month33=numeric(),
Month34=numeric(),
Month35=numeric(),
Month36=numeric(),
Month37=numeric(),
Month38=numeric(),
Month39=numeric(),
Month40=numeric(),
Month41=numeric(),
Month42=numeric(),
Month43=numeric(),
Month44=numeric(),
Month45=numeric(),
Month46=numeric(),
Month47=numeric(),
Month48=numeric(),
Month49=numeric(),
Month50=numeric(),
Month51=numeric(),
Month52=numeric(),
Month53=numeric(),
Month54=numeric(),

```
  stringsAsFactors=FALSE)
dfactualrequirements=data.frame(
  Month1=numeric(),
  Month2=numeric(),
  Month3=numeric(),
  Month4=numeric(),
  Month5=numeric(),
  Month6=numeric(),
  Month7=numeric(),
  Month8=numeric(),
  Month9=numeric(),
  Month10=numeric(),
  Month11=numeric(),
  Month12=numeric(),
  Month13=numeric(),
  Month14=numeric(),
  Month15=numeric(),
  Month16=numeric(),
  Month17=numeric(),
  Month18=numeric(),
  Month19=numeric(),
  Month20=numeric(),
  Month21=numeric(),
  Month22=numeric(),
  Month23=numeric(),
  Month24=numeric(),
  Month25=numeric(),
  Month26=numeric(),
  Month27=numeric(),
  Month28=numeric(),
  Month29=numeric(),
  Month30=numeric(),
  Month31=numeric(),
  Month32=numeric(),
  Month33=numeric(),
  Month34=numeric(),
  Month35=numeric(),
  Month36=numeric(),
  Month37=numeric(),
  Month38=numeric(),
  Month39=numeric(),
  Month40=numeric(),
  Month41=numeric(),
  Month42=numeric(),
  Month43=numeric(),
```

```
  Month44=numeric(),
  Month45=numeric(),
  Month46=numeric(),
  Month47=numeric(),
  Month48=numeric(),
  Month49=numeric(),
  Month50=numeric(),
  Month51=numeric(),
  Month52=numeric(),
  Month53=numeric(),
  Month54=numeric(),
  stringsAsFactors=FALSE)
dfbyNIIN=data.frame(
  NIIN=(levels(reqs$NIIN)),
  stringsAsFactors = FALSE)
####set up number of replications of experiment:####
NumberReps=30
#Configure by NIIN data collection to have a column for each replication
NewNIINcols=1
while(NewNIINcols<=NumberReps){
  dfbyNIIN[,(1+NewNIINcols)]=NA
  NewNIINcols=NewNIINcols+1
}
#initialize experiment
RepNum=1
start=Sys.time()
while(RepNum<=NumberReps){
#loop through all NIINs
for ( j in levels(reqs$NIIN) ) {
#subset out all NIINs except for the current one
NIIN=reqs[reqs$NIIN==j,]
#set up event queue data frame for current NIIN
EQ=NIIN
EQ$Reset=0
EQ$Fill=0
EQ$BB=0
EQ$NewEOQ=0
EQ$NewROP=0
EQ$QinBB=0
EQ$LenBB=0
EQ$NIIN=NULL
EQ=EQ[,c(1,3,2,4,5,6,7,8,9)]
colnames(EQ)[3]="DeltaQ"
EQ$DeltaQ=as.numeric(EQ$DeltaQ)
EQ$DeltaQ=-1*EQ$DeltaQ
```

```
EQ$LT=0
####adjust initialization date based on start of simulation####
#initialize date at start of simulation period
t=as.Date("2012-09-30")
####add additional WIOM output files to event q here####
#pull WIOM output information for current NIIN
WIOM_item=WIOM_1210[WIOM_1210$NIIN==j,]
#set EOQ,ROP, and LT at start of simulation
EOQ=max(1,round(WIOM_item[,2]))
ROP=WIOM_item[,3]
LT=round_any((WIOM_item[,4]*90),1,ceiling)
#make new row in event queue for WIOM output file info update
EQ[nrow(EQ)+1,]=EQ[nrow(EQ),]
EQ[nrow(EQ),]$Date=as.Date("2012-09-30")
EQ[nrow(EQ),][,c(2:9)]=0
EQ[nrow(EQ),]$Reset=1
#populate event queue with WIOM output info
EQ[nrow(EQ),]$NewEOQ=max(1,round(WIOM_item[,2]))
EQ[nrow(EQ),]$NewROP=WIOM_item[,3]
EQ[nrow(EQ),]$LT=round_any((WIOM_item[,4]*90),1,ceiling)
#order event q by date
EQ=EQ[order(EQ[,1]),]
row.names(EQ)=1:nrow(EQ)
#add next WIOM output to event q
WIOM_item=WIOM_1301[WIOM_1301$NIIN==j,]
#make new row in event queue for WIOM output info update
EQ[nrow(EQ)+1,]=EQ[nrow(EQ),]
EQ[nrow(EQ),]$Date=as.Date("2013-01-01")
EQ[nrow(EQ),][,c(2:9)]=0
EQ[nrow(EQ),]$Reset=1
#populate event queue with WIOM output info
EQ[nrow(EQ),]$NewEOQ=max(1,round(WIOM_item[,2]))
EQ[nrow(EQ),]$NewROP=WIOM_item[,3]
EQ[nrow(EQ),]$LT=round_any((WIOM_item[,4]*90),1,ceiling)
#order event q by date
EQ=EQ[order(EQ[,1]),]
row.names(EQ)=1:nrow(EQ)
#add next WIOM output to event q
WIOM_item=WIOM_1304[WIOM_1304$NIIN==j,]
#make new row in event queue for WIOM output info update
EQ[nrow(EQ)+1,]=EQ[nrow(EQ),]
EQ[nrow(EQ),]$Date=as.Date("2013-04-01")
EQ[nrow(EQ),][,c(2:9)]=0
EQ[nrow(EQ),]$Reset=1
#populate event queue with WIOM output info
```

```
EQ[nrow(EQ),]$NewEOQ=max(1,round(WIOM_item[,2]))
EQ[nrow(EQ),]$NewROP=WIOM_item[,3]
EQ[nrow(EQ),]$LT=round_any((WIOM_item[,4]*90),1,ceiling)
#order event q by date
EQ=EQ[order(EQ[,1]),]
row.names(EQ)=1:nrow(EQ)
#add next WIOM output to event q
WIOM_item=WIOM_1307[WIOM_1307$NIIN==j,]
#make new row in event queue for WIOM output info update
EQ[nrow(EQ)+1,]=EQ[nrow(EQ),]
EQ[nrow(EQ),]$Date=as.Date("2013-07-01")
EQ[nrow(EQ),][,c(2:9)]=0
EQ[nrow(EQ),]$Reset=1
#populate event queue with WIOM output info
EQ[nrow(EQ),]$NewEOQ=max(1,round(WIOM_item[,2]))
EQ[nrow(EQ),]$NewROP=WIOM_item[,3]
EQ[nrow(EQ),]$LT=round_any((WIOM_item[,4]*90),1,ceiling)
#order event q by date
EQ=EQ[order(EQ[,1]),]
row.names(EQ)=1:nrow(EQ)
#add next WIOM output to event q
WIOM_item=WIOM_1310[WIOM_1310$NIIN==j,]
#make new row in event queue for WIOM output info update
EQ[nrow(EQ)+1,]=EQ[nrow(EQ),]
EQ[nrow(EQ),]$Date=as.Date("2013-10-01")
EQ[nrow(EQ),][,c(2:9)]=0
EQ[nrow(EQ),]$Reset=1
#populate event queue with WIOM output info
EQ[nrow(EQ),]$NewEOQ=max(1,round(WIOM_item[,2]))
EQ[nrow(EQ),]$NewROP=WIOM_item[,3]
EQ[nrow(EQ),]$LT=round_any((WIOM_item[,4]*90),1,ceiling)
#order event q by date
EQ=EQ[order(EQ[,1]),]
row.names(EQ)=1:nrow(EQ)
#add next WIOM output to event q
WIOM_item=WIOM_1401[WIOM_1401$NIIN==j,]
#make new row in event queue for WIOM output info update
EQ[nrow(EQ)+1,]=EQ[nrow(EQ),]
EQ[nrow(EQ),]$Date=as.Date("2014-01-01")
EQ[nrow(EQ),][,c(2:9)]=0
EQ[nrow(EQ),]$Reset=1
#populate event queue with WIOM output info
EQ[nrow(EQ),]$NewEOQ=max(1,round(WIOM_item[,2]))
EQ[nrow(EQ),]$NewROP=WIOM_item[,3]
EQ[nrow(EQ),]$LT=round_any((WIOM_item[,4]*90),1,ceiling)
```

```
#order event q by date
EQ=EQ[order(EQ[,1]),]
row.names(EQ)=1:nrow(EQ)
#add next WIOM output to event q
WIOM_item=WIOM_1404[WIOM_1404$NIIN==j,]
#make new row in event queue for WIOM output info update
EQ[nrow(EQ)+1,]=EQ[nrow(EQ),]
EQ[nrow(EQ),]$Date=as.Date("2014-04-01")
EQ[nrow(EQ),][,c(2:9)]=0
EQ[nrow(EQ),]$Reset=1
#populate event queue with WIOM output info
EQ[nrow(EQ),]$NewEOQ=max(1,round(WIOM_item[,2]))
EQ[nrow(EQ),]$NewROP=WIOM_item[,3]
EQ[nrow(EQ),]$LT=round_any((WIOM_item[,4]*90),1,ceiling)
#order event q by date
EQ=EQ[order(EQ[,1]),]
row.names(EQ)=1:nrow(EQ)
#add next WIOM output to event q
WIOM_item=WIOM_1407[WIOM_1407$NIIN==j,]
#make new row in event queue for WIOM output info update
EQ[nrow(EQ)+1,]=EQ[nrow(EQ),]
EQ[nrow(EQ),]$Date=as.Date("2014-07-01")
EQ[nrow(EQ),][,c(2:9)]=0
EQ[nrow(EQ),]$Reset=1
#populate event queue with WIOM output info
EQ[nrow(EQ),]$NewEOQ=max(1,round(WIOM_item[,2]))
EQ[nrow(EQ),]$NewROP=WIOM_item[,3]
EQ[nrow(EQ),]$LT=round_any((WIOM_item[,4]*90),1,ceiling)
#order event q by date
EQ=EQ[order(EQ[,1]),]
row.names(EQ)=1:nrow(EQ)
#pull WIOM output information for current NIIN
WIOM_item=WIOM_1410[WIOM_1410$NIIN==j,]
#make new row in event queue for WIOM output file info update
EQ[nrow(EQ)+1,]=EQ[nrow(EQ),]
EQ[nrow(EQ),]$Date=as.Date("2014-10-01")
EQ[nrow(EQ),][,c(2:9)]=0
EQ[nrow(EQ),]$Reset=1
#populate event queue with WIOM output info
EQ[nrow(EQ),]$NewEOQ=max(1,round(WIOM_item[,2]))
EQ[nrow(EQ),]$NewROP=WIOM_item[,3]
EQ[nrow(EQ),]$LT=round_any((WIOM_item[,4]*90),1,ceiling)
#order event q by date
EQ=EQ[order(EQ[,1]),]
row.names(EQ)=1:nrow(EQ)
```

```
#add next WIOM output to event q
WIOM_item=WIOM_1501[WIOM_1501$NIIN==j,]
#make new row in event queue for WIOM output info update
EQ[nrow(EQ)+1,]=EQ[nrow(EQ),]
EQ[nrow(EQ),]$Date=as.Date("2015-01-01")
EQ[nrow(EQ),][,c(2:9)]=0
EQ[nrow(EQ),]$Reset=1
#populate event queue with WIOM output info
EQ[nrow(EQ),]$NewEOQ=max(1,round(WIOM_item[,2]))
EQ[nrow(EQ),]$NewROP=WIOM_item[,3]
EQ[nrow(EQ),]$LT=round_any((WIOM_item[,4]*90),1,ceiling)
#order event q by date
EQ=EQ[order(EQ[,1]),]
row.names(EQ)=1:nrow(EQ)
#add next WIOM output to event q
WIOM_item=WIOM_1504[WIOM_1504$NIIN==j,]
#make new row in event queue for WIOM output info update
EQ[nrow(EQ)+1,]=EQ[nrow(EQ),]
EQ[nrow(EQ),]$Date=as.Date("2015-04-01")
EQ[nrow(EQ),][,c(2:9)]=0
EQ[nrow(EQ),]$Reset=1
#populate event queue with WIOM output info
EQ[nrow(EQ),]$NewEOQ=max(1,round(WIOM_item[,2]))
EQ[nrow(EQ),]$NewROP=WIOM_item[,3]
EQ[nrow(EQ),]$LT=round_any((WIOM_item[,4]*90),1,ceiling)
#order event q by date
EQ=EQ[order(EQ[,1]),]
row.names(EQ)=1:nrow(EQ)
#add next WIOM output to event q
WIOM_item=WIOM_1507[WIOM_1507$NIIN==j,]
#make new row in event queue for WIOM output info update
EQ[nrow(EQ)+1,]=EQ[nrow(EQ),]
EQ[nrow(EQ),]$Date=as.Date("2015-07-01")
EQ[nrow(EQ),][,c(2:9)]=0
EQ[nrow(EQ),]$Reset=1
#populate event queue with WIOM output info
EQ[nrow(EQ),]$NewEOQ=max(1, round(WIOM_item[,2]))
EQ[nrow(EQ),]$NewROP=WIOM_item[,3]
EQ[nrow(EQ),]$LT=round_any((WIOM_item[,4]*90),1,ceiling)
#order event q by date
EQ=EQ[order(EQ[,1]),]
row.names(EQ)=1:nrow(EQ)
#add next WIOM output to event q
WIOM_item=WIOM_1510[WIOM_1510$NIIN==j,]
```

```
#make new row in event queue for WIOM output info update
EQ[nrow(EQ)+1,]=EQ[nrow(EQ),]
EQ[nrow(EQ),]$Date=as.Date("2015-10-01")
EQ[nrow(EQ),][,c(2:9)]=0
EQ[nrow(EQ),]$Reset=1
#populate event queue with WIOM output info
EQ[nrow(EQ),]$NewEOQ=max(1,round(WIOM_item[,2]))
EQ[nrow(EQ),]$NewROP=WIOM_item[,3]
EQ[nrow(EQ),]$LT=round_any((WIOM_item[,4]*90),1,ceiling)
#order event q by date
EQ=EQ[order(EQ[,1]),]
row.names(EQ)=1:nrow(EQ)
#add next WIOM output to event q
WIOM_item=WIOM_1601[WIOM_1601$NIIN==j,]
#make new row in event queue for WIOM output info update
EQ[nrow(EQ)+1,]=EQ[nrow(EQ),]
EQ[nrow(EQ),]$Date=as.Date("2016-01-01")
EQ[nrow(EQ),][,c(2:9)]=0
EQ[nrow(EQ),]$Reset=1
#populate event queue with WIOM output info
EQ[nrow(EQ),]$NewEOQ=max(1,round(WIOM_item[,2]))
EQ[nrow(EQ),]$NewROP=WIOM_item[,3]
EQ[nrow(EQ),]$LT=round_any((WIOM_item[,4]*90),1,ceiling)
#order event q by date
EQ=EQ[order(EQ[,1]),]
row.names(EQ)=1:nrow(EQ)
#add next WIOM output to event q
WIOM_item=WIOM_1604[WIOM_1604$NIIN==j,]
#make new row in event queue for WIOM output info update
EQ[nrow(EQ)+1,]=EQ[nrow(EQ),]
EQ[nrow(EQ),]$Date=as.Date("2016-04-01")
EQ[nrow(EQ),][,c(2:9)]=0
EQ[nrow(EQ),]$Reset=1
#populate event queue with WIOM output info
EQ[nrow(EQ),]$NewEOQ=max(1,round(WIOM_item[,2]))
EQ[nrow(EQ),]$NewROP=WIOM_item[,3]
EQ[nrow(EQ),]$LT=round_any((WIOM_item[,4]*90),1,ceiling)
#order event q by date
EQ=EQ[order(EQ[,1]),]
row.names(EQ)=1:nrow(EQ)
#add next WIOM output to event q
WIOM_item=WIOM_1607[WIOM_1607$NIIN==j,]
#make new row in event queue for WIOM output info update
EQ[nrow(EQ)+1,]=EQ[nrow(EQ),]
EQ[nrow(EQ),]$Date=as.Date("2016-07-01")
```

```
EQ[nrow(EQ),][,c(2:9)]=0
EQ[nrow(EQ),]$Reset=1
#populate event queue with WIOM output info
EQ[nrow(EQ),]$NewEOQ=max(1,round(WIOM_item[,2]))
EQ[nrow(EQ),]$NewROP=WIOM_item[,3]
EQ[nrow(EQ),]$LT=round_any((WIOM_item[,4]*90),1,ceiling)
#order event q by date
EQ=EQ[order(EQ[,1]),]
row.names(EQ)=1:nrow(EQ)
#add next WIOM output to event q
WIOM_item=WIOM_1610[WIOM_1610$NIIN==j,]
#make new row in event queue for WIOM output info update
EQ[nrow(EQ)+1,]=EQ[nrow(EQ),]
EQ[nrow(EQ),]$Date=as.Date("2016-10-01")
EQ[nrow(EQ),][,c(2:9)]=0
EQ[nrow(EQ),]$Reset=1
#populate event queue with WIOM output info
EQ[nrow(EQ),]$NewEOQ=max(1,round(WIOM_item[,2]))
EQ[nrow(EQ),]$NewROP=WIOM_item[,3]
EQ[nrow(EQ),]$LT=round_any((WIOM_item[,4]*90),1,ceiling)
#order event q by date
EQ=EQ[order(EQ[,1]),]
row.names(EQ)=1:nrow(EQ)
#add next WIOM output to event q
WIOM_item=WIOM_1701[WIOM_1701$NIIN==j,]
#make new row in event queue for WIOM output info update
EQ[nrow(EQ)+1,]=EQ[nrow(EQ),]
EQ[nrow(EQ),]$Date=as.Date("2017-01-01")
EQ[nrow(EQ),][,c(2:9)]=0
EQ[nrow(EQ),]$Reset=1
#populate event queue with WIOM output info
EQ[nrow(EQ),]$NewEOQ=max(1,round(WIOM_item[,2]))
EQ[nrow(EQ),]$NewROP=WIOM_item[,3]
EQ[nrow(EQ),]$LT=round_any((WIOM_item[,4]*90),1,ceiling)
#order event q by date
EQ=EQ[order(EQ[,1]),]
row.names(EQ)=1:nrow(EQ)
#initialize stock posture at beginning of simulation
OutstandingReorder=0
#bootstrap starting condition of quantity o/h
#random uniform integer between 0 and target inventory
Q=sample(1:(EOQ),1)
if(ROP<0){
  Q=0
}
```

```
#initialize inventory position equal to quantity o/h
IP=Q
#initialize outstanding reorders at start of simulation
#while IP is less than ROP, add a reorder to event q
while(IP<=ROP){
 OutstandingReorder=OutstandingReorder+1
 IP=IP+EOQ
 EQ[nrow(EQ)+1,]=EQ[nrow(EQ),]
 EQ[nrow(EQ),]$Date=t+round_any((Q/ROP)*LT,1)
 EQ[nrow(EQ),][,c(2:9)]=0
 EQ[nrow(EQ),]$DeltaQ=EOQ
 EQ=EQ[order(EQ[,1]),]
 row.names(EQ)=1:nrow(EQ)
}
#loop through all events in event q for current NIIN
i=1
while(i<nrow(EQ)+1){
 #update time counter
 t=EQ[i,]$Date
 #check to see if event is a parameter update
 #if yes, update EOQ, ROP, Lead Time
 if(EQ[i,]$Reset==1){
  EOQ=EQ[i,]$NewEOQ
  ROP=EQ[i,]$NewROP
  LT=EQ[i,]$LT
 }
 #check to see if event is a requisition
 #if yes, perform requisition tasks
 if(EQ[i,]$DeltaQ<0){
  #check to see if quantity o/h can fill req
  #if it can't, flag event row as a backorder
  #and calculate the quantity in backorder status
  if(abs(EQ[i,]$DeltaQ)>Q){
   EQ[i,]$BB=1
   if(Q>0){
    EQ[i,]$QinBB=abs(Q+EQ[i,]$DeltaQ)
   }
   if(Q<=0){
    EQ[i,]$QinBB=abs(EQ[i,]$DeltaQ)
   }
  }
  #if quantity o/h can satisfy req,
  #flag event row as a filled req
  if(abs(EQ[i,]$DeltaQ)<=Q){
   EQ[i,]$Fill=1
```

```
  }
 }
 #update quantity o/h
 Q=Q+EQ[i,]$DeltaQ
 #update inventory position
 #inventory position only changes for reqs,
 #arriving reorders don't change it
 if (EQ[i,]$DeltaQ<0){
  IP=IP+EQ[i,]$DeltaQ
 }
 #check if event is an arriving reorder
 #if yes, perform reorder actions
 if(EQ[i,]$DeltaQ>0){
  #reduce the amount of outstanding reorders
  #to reflect that one just came in
  OutstandingReorder=OutstandingReorder-1
  #record the quantity arriving
  ReorderQuantity=EQ[i,]$DeltaQ
  #make a vector of all reqs that have outstanding
  #quantity in backorder
  BBvec=which((EQ$QinBB>0))
  #cycle through each req in event q with quantity in backorder,
  for ( k in BBvec ){
   #if amount in reorder is not enough to satisfy backorder
   #reduce quantity in backorder by available reorder
   #and set available reorder to 0
   if (ReorderQuantity<EQ[k,]$QinBB){
    EQ[k,]$QinBB=EQ[k,]$QinBB-ReorderQuantity
    ReorderQuantity=0
   }
   #if amount in reorder is enough to satisfy backorder
   #reduce amount of available reorder quantity,
   #and set the quantity in backorder for that req to 0
   #record the length of the backorder
   if (ReorderQuantity>=EQ[k,]$QinBB){
    ReorderQuantity=ReorderQuantity-EQ[k,]$QinBB
    EQ[k,]$QinBB=0
    EQ[k,]$LenBB=as.numeric(as.Date(EQ[i,]$Date))-
as.numeric(as.Date(EQ[k,]$Date))
   }
  }
 }
 #at completion of event actions, check IP against ROP
 #and add reorder to event q if necessary
 while(IP<=ROP){
```

```
    OutstandingReorder=OutstandingReorder+1
    IP=IP+EOQ
    EQ[nrow(EQ)+1,]=EQ[nrow(EQ),]
    EQ[nrow(EQ),]$Date=t+LT*24*60*60
    EQ[nrow(EQ),][,c(2:9)]=0
    EQ[nrow(EQ),]$DeltaQ=EOQ
    EQ=EQ[order(EQ[,1]),]
    row.names(EQ)=1:nrow(EQ)
  }
  i=i+1
###End of i loop###
}
#make dataframe that collects performance info for this NIIN
RECAP_item=EQ[((EQ$Fill==1)|(EQ$BB==1)),]
RECAP_item=RECAP_item[,c(1,4,5,9)]
#combine dataframe for this NIIN with all others
RECAP_total=rbind(RECAP_item,RECAP_total)
#record by NIIN data
NIIN_rate=sum(RECAP_item$Fill)/nrow(RECAP_item)
dfbyNIIN[(dfbyNIIN$NIIN==j),(1+RepNum)]=NIIN_rate
}
###End of j loop###
###Collect Data on this replication of the experiment:
#order RECAP by date
RECAP_total=RECAP_total[order(RECAP_total$Date),]
#Assign a tag to each req saying what month it happened
RECAP_total$increments=cut.POSIXt(RECAP_total$Date, breaks="month")
#sum up all BBs and Fills in each month, combine into a single dataframe
output.df=aggregate(x=RECAP_total$Fill, by = list(time.increment =
RECAP_total$increments),FUN=sum, na.rm=TRUE)
output2.df=aggregate(x=RECAP_total$BB, by = list(time.increment =
RECAP_total$increments),FUN=sum, na.rm=TRUE)
output3.df=cbind(output.df,output2.df)
#calculate the fillrate for each month
output3.df$fillrate=output3.df[,2]/(output3.df[,2]+output3.df[,4])
#populate summary df with fill rate info for each month
p=1
while (p<=length(unique(RECAP_total$increments))){
  dfsum[RepNum,p]=output3.df[p,]$fillrate
  p=p+1
}
#populate summary df with total fill rate info and average BB length for entire time period
dfsum[RepNum,]$TotalFillRate=mean(RECAP_total$Fill)
dfsum[RepNum,]$AvgBB=mean(RECAP_total[(RECAP_total$LenBB>0),]$LenBB)
```

```
####Info by months (Raw number of fills)
#populate summary df with fill info for each month
p=1
while (p<=length(unique(RECAP_total$increments))){
  dfactualfills[RepNum,p]=output3.df[p,2]
  p=p+1
}
####Info by months (Raw number of requirements)
#populate summary df with fill info for each month
p=1
while (p<=length(unique(RECAP_total$increments))){
  dfactualrequirements[RepNum,p]=output3.df[p,2]+output3.df[p,4]
  p=p+1
}
#clean up df's to save memory
RECAP_total=NULL
output.df=NULL
output2.df=NULL
output3.df=NULL
if(RepNum==2){
  write.csv(dfsum, file = "ExperimentRates2.csv")
  write.csv(dfactualfills, file = "ExperimentFills2.csv")
  write.csv(dfactualrequirements, file = "ExperimentRequirements2.csv")
  write.csv(dfbyNIIN, file = "ExperimentNIINbreakDown2.csv")
}
if(RepNum==8){
  write.csv(dfsum, file = "ExperimentRates8.csv")
  write.csv(dfactualfills, file = "ExperimentFills8.csv")
  write.csv(dfactualrequirements, file = "ExperimentRequirements8.csv")
  write.csv(dfbyNIIN, file = "ExperimentNIINbreakDown8.csv")
}
if(RepNum==15){
  write.csv(dfsum, file = "ExperimentRates15.csv")
  write.csv(dfactualfills, file = "ExperimentFills15.csv")
  write.csv(dfactualrequirements, file = "ExperimentRequirements15.csv")
  write.csv(dfbyNIIN, file = "ExperimentNIINbreakDown15.csv")
}
if(RepNum==20){
  write.csv(dfsum, file = "ExperimentRates20.csv")
  write.csv(dfactualfills, file = "ExperimentFills20.csv")
  write.csv(dfactualrequirements, file = "ExperimentRequirements20.csv")
  write.csv(dfbyNIIN, file = "ExperimentNIINbreakDown20.csv")
}
#next replication
RepNum=RepNum+1
```

```
}
finish=Sys.time()
length=finish-start
print(length)
####end of experiment####
write.csv(dfsum, file = "ExperimentRates30.csv")
write.csv(dfactualfills, file = "ExperimentFills30.csv")
write.csv(dfactualrequirements, file = "ExperimentRequirements30.csv")
write.csv(dfbyNIIN, file = "ExperimentNIINbreakDown30.csv")
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Ellis D, Cardillo J, Motter, M (2017) Information regarding NAVSUP WSS operations and WIOM implementation provided to author in person via personal communication during site visit to NAVSUP WSS, June 13–15.

Ellis D (2017) Data files from NAVSUP WSS provided to author via personal communication, October 18–19.

Law A (2015) *Simulation Modeling and Analysis*, 5th ed. (McGraw-Hill Education, New York, NY).

Motter M (2017) Data files from NAVSUP WSS provided to author via personal communication, December 20.

NAVSUP (2018) NAVSUP Enterprise. Accessed February 11, 2017, https://www.navsup.navy.mil/public/navsup/enterprise/.

R Core Team (2016) R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/.

Roth G (2016) A simulation of alternative for wholesale inventory replenishment. Master's thesis, Operations Research Department, Naval Postgraduate School, Monterey, CA. https://calhoun.nps.edu/handle/10945/48587.

Salmeron J, Craparo E (2017) *Wholesale Inventory Optimization Model.* Naval Postgraduate School, Monterey, CA.

Silver E, Pyke D, Peterson R (1998) *Inventory Management and Production Planning and Scheduling*, 3rd ed. (John Wiley and Sons, Hoboken, NJ).

Tersine R (1994) *Principles of Inventory and Materials Management*, 4th ed. (PTR Prentice Hall, Upper Saddle River, NJ).

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California