2018-06

# USMC DEPOT-LEVEL MAINTENANCE OF THE LIGHT ARMORED VEHICLE (LAV): A DISCRETE-EVENT SIMULATION ANALYSIS

Blankenbeker, Michael J.

Monterey, CA; Naval Postgraduate School

THESIS

# USMC DEPOT-LEVEL MAINTENANCE OF THE LIGHT ARMORED VEHICLE (LAV): A DISCRETE-EVENT SIMULATION ANALYSIS

by

Michael J. Blankenbeker

June 2018

| | |
|---|---|
| Thesis Advisor: | Arnold H. Buss |
| Second Reader: | Ruriko Yoshida |

**This thesis was performed at the MOVES Institute.**

| REPORT DOCUMENTATION PAGE | *Form Approved OMB No. 0704-0188* |
|---|---|

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

| | |
|---|---|
| **4. TITLE AND SUBTITLE** USMC DEPOT-LEVEL MAINTENANCE OF THE LIGHT ARMORED VEHICLE (LAV): A DISCRETE-EVENT SIMULATION ANALYSIS | **5. FUNDING NUMBERS** |
| **6. AUTHOR(S)** Michael J. Blankenbeker | |
| **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(E**S) 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.

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

**13. ABSTRACT (maximum 200 words)**

The USMC maintenance depots at Marine Corps Logistics Base (MCLB) Albany, GA, and MCLB Barstow, CA, conduct extensive contracted overhauls and repairs on a variety of ground combat and combat service support vehicles from throughout the operating forces. Servicing of one vehicle in particular, the Light Armored Vehicle (LAV), has had issues with runaway costs and prolonged maintenance cycle-time caused by severe bottlenecks at key junctures in the maintenance cycle.

This study models the bottlenecks experienced in the real system and then provides recommendations to mitigate them. The discrete-event simulation (DES) tools used in this study implement data farming and data analysis that provide quantitative justification and show the sponsor where to adjust resource capacity parameters in the system in order to reduce the effect of these bottlenecks and overall cycle time. In addition to the DES analysis, this project provides the sponsor, Marine Corps Logistics Command, with a working tool that can be used in assisting key leadership in making resource capacity decisions by showing how individual queues and the overall system are affected when input parameters are adjusted.

| | |
|---|---|
| **14. SUBJECT TERMS** discrete event simulation, Light Armored Vehicle, Marine Corps Logistics Command, Marine Depot Maintenance Command | **15. NUMBER OF PAGES** 73 |
| | **16. PRICE CODE** |

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

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

# USMC DEPOT-LEVEL MAINTENANCE OF THE LIGHT ARMORED VEHICLE (LAV): A DISCRETE-EVENT SIMULATION ANALYSIS

Michael J. Blankenbeker
Captain, United States Marine Corps
BS, Georgia Institute of Technology, 2011

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN**
**MODELING, VIRTUAL ENVIRONMENTS, AND SIMULATION**

from the

**NAVAL POSTGRADUATE SCHOOL**
**June 2018**

Approved by: Arnold H. Buss
Advisor

Ruriko Yoshida
Second Reader

Peter J. Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

# ABSTRACT

The USMC maintenance depots at Marine Corps Logistics Base (MCLB) Albany, GA, and MCLB Barstow, CA, conduct extensive contracted overhauls and repairs on a variety of ground combat and combat service support vehicles from throughout the operating forces. Servicing of one vehicle in particular, the Light Armored Vehicle (LAV), has had issues with runaway costs and prolonged maintenance cycle-time caused by severe bottlenecks at key junctures in the maintenance cycle.

This study models the bottlenecks experienced in the real system and then provides recommendations to mitigate them. The discrete-event simulation (DES) tools used in this study implement data farming and data analysis that provide quantitative justification and show the sponsor where to adjust resource capacity parameters in the system in order to reduce the effect of these bottlenecks and overall cycle time. In addition to the DES analysis, this project provides the sponsor, Marine Corps Logistics Command, with a working tool that can be used in assisting key leadership in making resource capacity decisions by showing how individual queues and the overall system are affected when input parameters are adjusted.

THIS PAGE INTENTIONALLY LEFT BLANK

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF ACRONYMS AND ABBREVIATIONS

| | |
|---|---|
| AAV | Assault Amphibian Vehicle |
| LAV | Light Armored Vehicle |
| LOGCOM | Marine Corps Logistics Command |
| MAGTF | Marine Air Ground Task Force |
| MCLB | Marine Corps Logistics Base |
| MDMC | Marine Depot Maintenance Command |
| MOVES | Modeling, Virtual Environments, and Simulation |
| IROAN | Inspect and Repair Only as Necessary |
| SLEP | Service Life Extension Program |
| LTI | Limited Technical Inspection |
| DES | Discrete Event Simulation |
| NaN | Not a Number |
| HDT | Heat Distortion Temperature |
| FIFO | First In First Out |
| ANOVA | Analysis of Variance |
| HSD | Honest Significant Difference |
| NOLH | Nearly Orthogonal Latin Hypercube |
| SEED | Simulation, Experiments, and Efficient Design |
| VV&A | Verification, Validation, and Accreditation |

THIS PAGE INTENTIONALLY LEFT BLANK

# ACKNOWLEDGMENTS

Completing this work was no small endeavor, and there are many who contributed to my success. I would like to thank the NPS staff who assisted me in this effort, Dr. Arnold Buss and Dr. Ruriko Yoshida. Without their expertise and counsel, this project would have been markedly more difficult. I would also like to thank the personnel at the Marine Corps Logistics Command, Mr. Huntley Bodden, Major Brian Bagley, and Mr. Kyle Luckie, for their support and assistance in the data collection and scoping of this project. Most importantly, I would like to thank my wife, Angela, for her continuous support throughout my pursuit of this achievement and in my career.

THIS PAGE INTENTIONALLY LEFT BLANK

# I. INTRODUCTION

In the 1980s, as a response to an emerging need to have a rapidly deployable, highly maneuverable capability that could address threats in an urban environment as well rugged expeditionary environments, the United States Marine Corps developed and purchased the Light Armored Vehicle (LAV) program (Mullins, Adams, & Simms, 2005). Initially fielded in 1986 with a 20-year expected life span, the LAV has been a mainstay of the Marine Corps' ground combat arsenal (Mullins et al., 2005). The nature of the strain that the LAV fleet has undergone while traversing rugged terrain in both training and operational environments necessitate an incredibly immense maintenance effort.

Maintenance in the Marine Corps is a rather structured effort in which levels of responsibility for completed different types of work are delineated into echelons and tasked the various supporting agencies within the operating forces. 1st Echelon maintenance is also known as operator level maintenance, and is conducted by the unit or crew that owns the equipment and uses it for their assigned mission set ("Marine Corps Order 4790.2C," 2012). Examples of 1st echelon maintenance would be weekly routine vehicle checks and inspections such as checking fluid levels, tire wear, presence of any fluid leaks, or damage to vehicle parts or issued equipment the operator is responsible for maintaining. If the operator identifies concerns that are outside their capability to address, they will induct the equipment in for 2nd echelon maintenance ("Marine Corps Order 4790.2C," 2012). 2nd echelon maintenance is conducted by personnel that are trained mechanics on that given piece of equipment ("Marine Corps Order 4790.2C," 2012).

2nd echelon maintenance would include repairs that do not involve pulling major drivetrain components off of the vehicle such as hose replacements, electrical component replacements, or routine semi-annual or annual preventive maintenance such as oil changes. 1st and 2nd echelon maintenance are together also known as the organic level of maintenance, because they consist of repairs conducted by personnel within the unit that owns and uses the equipment ("Marine Corps Order 4790.2C," 2012).

If a particular repair is above the capability of that unit’s maintenance section, then it must be inducted into the 3rd echelon maintenance shop (“Marine Corps Order 4790.2C,” 2012). If that particular battalion has a third echelon maintenance capability, then it could be that the company’s mechanics will induct the equipment into the battalion’s maintenance section for that higher level repair. You may see this transaction most often at units who are typically the sole custodian of that piece of equipment on the base, such as a tank battalion or Assault Amphibian Vehicle (AAV) Battalion. For equipment that is held and maintained by a variety of units on that particular installation, such as motor transport equipment, the battalion that owns the equipment will induct the equipment into their designated direct support maintenance activity within the Marine Logistics Group. The types of work done at 3rd echelon might be more complex scheduled preventive maintenance and the removal and replacement of major component items such as transmissions and engines.

While the major components are replaced at the 3rd echelon shop, those component items are pulled from the end item, packaged, shipped, and inducted at the 4th echelon shop for repair (“Marine Corps Order 4790.2C,” 2012). The 4th echelon shop has the primary responsibility of receipt, repair and/or rebuilding of major component items, and then return of fixed component item back to the 3rd echelon shop for re-installation into the end item (“Marine Corps Order 4790.2C,” 2012). The 4th echelon repairs are typically performed by maintenance Marines at the highest support agency within the Marine Air Ground Task Force (MAGTF), possibly in conjunction with defense contracting company representatives, if the end item has some warranty or maintenance contract that covers those repairs. 3rd and 4th echelon are together referred to as the intermediate level of maintenance (“Marine Corps Order 4790.2C,” 2012).

Echelons one through four comprise the methods by which equipment in the maintenance cycle while still remaining in the operating forces and still owned by the original using unit. Periodically, however, the MAGTF will need to cycle a piece of equipment out of the operating forces for a complete overhaul in order to conduct complete rebuild of the equipment and all component items to whatever extent is necessary. This level is known as the 5th echelon of maintenance, also called depot-level maintenance

(“Marine Corps Order 4790.2C,” 2012). This study will be looking at the depot-level maintenance processes for the LAV.

Depot level maintenance for the Marine Corps is overseen by the Marine Corps Logistics Command (LOGCOM) and the Marine Depot Maintenance Command (MDMC) (Marine Corps Logistics Command, 2017). Depot-level maintenance is performed at two sites—Marine Corps Logistics Base (MCLB) Albany, GA, and MCLB Barstow, CA, both of which are overseen by MDMC based at MCLB Albany (Marine Corps Logistics Command, 2017).

The motivation for the sponsor having this study completed was originally drawn from the recommendations from a report by the Penn State Applied Research Laboratory (B. Bagley, personal communication, 29 January 2017), a team from which was invited to MDMC to conduct a review and provide ideas as to how LOGCOM can achieve a more efficient operating environment (Bair et al., 2017). Specifically, the Penn State Team recommended LOGCOM pursue the development of a discrete-event simulation model in order to truly assess the throughput capabilities of the facility so that they can understand how the system can be expected to perform by applying probability distributions for arrivals and service times throughout the different stations and processes in depot level maintenance (Bair et al., 2017). This will ideally lead to an improved ability to make decisions regarding adjustments to the key drivers for the maintenance production process and how system resources can be manipulated to achieve desired effects on the system.

The objective of this study will be to create a discrete-event model of the LAV depot-level maintenance process and conduct simulation experiments with the model to see how system performance can be expected to change based on how resources allocations are changed due to decisions made governing the system.

This project presents several challenges, some of which are incumbent upon any equipment involved in depot-level maintenance and some of which are unique to the LAV. Depot-level maintenance can be particularly difficult to forecast due to the nature of how equipment is repaired (K. Luckie, personal communication, 29 January 2018). For any given item that is inducted into MDMC for depot-level maintenance, the types of jobs that will be completed and the degree of repair that will be required will be unique to that individual item (United States Marine Corps, 1992). There is an incredible amount of variability in the time it will take to complete each task as it will depend entirely on how damaged or worn the given component may be (K. Luckie, personal communication, 29 January 2018). An engine could take anywhere from several hours to over a week to repair (K. Luckie, personal communication, 29 January 2018). Similarly, to complete a welding job on a vehicle's hull and body could take anywhere from a number of hours to over a month (K. Luckie, personal communication, 29 January 2018).

Another complication that is pertinent to the LAV right now is the numerous life extension packages that have been directed by the acquisition program office for the equipment (K. Luckie, personal communication, 29 January, 2018). The LAV program initially had a program expected lifespan of 20 years, which would have put the original disposition date for the entire program in the mid-2000s (Mullins et al., 2005). Due to both the conflicts that have taken place as well as no sensible cost-effective alternative, the program office has directed life-extension modifications to the equipment as it comes to the depot for maintenance (K. Luckie, personal communication, 29 January 2018). These modification packages create another layer of variation that can be difficult to model, especially early into the cycle of completion, as you will not have the data to understand to what degree the modification package changes the expected service time of any given station. This variation is compounded even further when the maintenance production stations are being based not in data, but off of the staff estimates of subject matter experts (K. Luckie, personal communication, 29 January 2018).

There are several concerns that will limit the ability to produce a useful model. While LOGCOM has asked for a complete, beginning to end discrete-event model that maps production through all steps in the maintenance cycle, there is currently no data for actual service times for each step in the maintenance process (K. Luckie, personal communication, 29 January 2018). Every station in the model has been based on subject matter expert's estimate of the average time is takes to complete a given step (K. Luckie, personal communication, 29 January 2018). This will be problematic as the observed variations seen in the system through the model will not be as representative as the actual variation. That being said, the discrete-event model based on this limited information can still be useful to leadership to begin to understand where capacity restrictions may be expected to cause bottlenecks, how capacity expansion may be expected to relieve bottlenecks, and how adjustments to resources to address known problems in one part of the system may potentially create other problems elsewhere. This model will primarily look at the system resources of personnel and space during the experimentation phase in order to provide recommendations for decisions to the LOGCOM leadership.

THIS PAGE INTENTIONALLY LEFT BLANK

# II. BACKGROUND

## A. PREVIOUS MDMC STUDIES

### 1. DES Used at MDMC

In preparation for conducting this study, a comprehensive review was undertaken of a variety of publications, including student theses related to the LAV, depot-level maintenance, and DES as well as professional studies and journal articles related to the employment of discrete-event simulation in supply chain and logistics networks outside the military domain.

One of the first works reviewed was that of a previous MOVES student, Maj Timothy Curling. In his study, Major Curling completed an overview of the generic depot-level maintenance process; however, he did not narrow down on any one particular end item, or group of items his scope was more broad and had a different goal (Curling, 2016). In Major Curling’s project he used both DES and optimization techniques to help provide an improved method for ordering repair parts for the equipment that undergoes maintenance at the Logistics bases (Curling, 2016). Seeing how Major Curling was able to implement a DES/optimization model was helpful in building the framework for this study, however there are some fundamental differences in both the models and objectives between Major Curling’s project and the overarching model being attempted in this LAV depot process study.

Where Major Curling’s project had a focus on inventory control in depot-level maintenance (Curling, 2016), this study will be focused more on measuring system throughput and cycle time. In addition, while Major Curling looked at the generic depot-level maintenance process (Curling, 2016), this model will be a little more narrowly focused on a specific end item, the Light Armored vehicle. One particular item of note is the maintenance production model. DES models can be represented in the form of an event graph (Schruben, 1983; Buss, 1995). An event graph is simply a depiction used to demonstrate the behavior found in a particular DES model (Schruben, 1983; Buss, 1995). In Major Curling’s production event graph, he has 12 nodes (Curling, 2016). If each step in the LAV process were implemented, there would be 156 nodes in the event graph. He accomplished this by looking at broad brushstroke movements such as “Start Disassembly” or “Finish Assembly” rather than specific steps in the process such as “remove armor plating” or “final communications inspection”; the takeaway is that not every step must be fleshed out in thorough detail, that a DES model can incorporate aggregation and still be valid. Please see Chapter IV for examples of event graphs.

One of the challenges Maj Curling faced was obtaining worthwhile data from the maintenance facilities (T. Curling, personal communication, 20 September 2017); the same difficulties can reasonably be expected in this study also. This particular issue provided insight that the baseline model being developed for the LAV may need to find ways to work around on hand data being present to map the system. Ultimately Major Curling’s thesis was able to provide recommendations to LOGCOM for more efficient repair part requisition forecasting; the proof of concept is that a DES model developed from limited data can still provide value to the client, in this case the same client.

### 2. LAV Business Study

The 2005 Business school thesis on LAV depot-level maintenance by Mullins, Adams, and Simms took a look at the business aspects of the maintenance undergone at both the Albany and Barstow facilities (Mullins et al., 2005). While their analysis is a bit old and does not incorporate any simulation, it provides some very insightful information on both the LAV processes and the depots themselves.

One of the major changes the depots made 15–20 years ago was the implementation of the theory of constraints method of maintenance production (Mullins et al., 2005). This called from shifting from an “assembly line” type process to a “workstation” method (Mullins et al., 2005). This essentially means that instead of doing every single step in perfect order, the maintenance crews will perform any work they can at a given workstation given that they have the necessary resources on hand to do so; even if it is not the traditional or typical order in which the maintenance tasks are performed (Mullins et al., 2005).

The next major concepts discussed in the paper are the IROAN and SLEP programs. The service life extension program (SLEP) is essentially a set of that all end items of a particular model coming to the facility will undergo and a predetermined modification in order to extend the program life from its original end date; the marine corps uses this program as a cost effective alternative to acquisition of a new product (Mullins et al., 2005). The inspect and repair only as necessary (IROAN) is a program that will have the personnel conduct a thorough limited technical inspection (LTI) upon the equipment's arrival to the depot in order to determine which items need to be repaired or replaced on the vehicle (Mullins et al., 2005). IROAN has in effect helped the maintenance depots avoid taking time to disassemble, replace, and reassemble components that are still functional (Mullins et al., 2005).

The next interesting piece in the paper was the comparisons between various costs at the maintenance depots. At Albany, the labor costs tended to be less than Barstow, while Barstow had cheaper costs for repair parts (Mullins et al., 2005). Another significant cost was that of shipping from Marine Corps units in Hawaii, and Okinawa, Japan, to the maintenance facility in Barstow (Mullins et al., 2005). Where it costs \$600–\$1000 for vehicles on the east and west coast units to get to their respective maintenance facilities, it costs around \$6000 to get from Hawaii and nearly \$10,000 to get from Okinawa to their respective depot-level maintenance facility (Mullins et al., 2005). These points leave unanswered questions: do these cost disparities between Albany and Barstow still exist, and what are the causes?

While the goal of the previous study was different from the intent of the upcoming simulation study, the background and insight into the evolution of processes at the maintenance depots is invaluable and provides perspective that helps us understand the full picture. It also helps build the case for some level of aggregation into the final model. Drilling down to low might not only be wasting time, it could be limiting the model to a degree of granularity that is neither practical nor realistic.

## B. PREVIOUS DES WORK

### 1. DES and Value Networks

"Integration of discrete-event simulation and optimization for the design of value networks," by M. Schlegel, G. Brosig, A. Eckert, M. Jung, A. Polt, M. Sonnenschein and

C. Vogt, was an incredibly relevant resource. In this particular paper, the authors describe the application of discrete event simulation and optimization in generic value networks, but specifically those in which discrete decisions can be made that affect the policy of a given manufacturing or supply chain type network (Schlegel et al., 2006). The application mentioned in this article provides an excellent proof of concept and basis for a discrete event simulation model which can test policies at a maintenance facility. This paper provides an excellent foundation and outline for how to create a model for resource capacity expansion in a given system (Schlegel et al., 2006). It serves as an excellent example of how a pool of resources can be allocated and tested at various points in a given network to examine effectiveness.

### 2. DES Application in Supply Chains

"Linking Supply Chain Configuration to Supply Chain Performance: a DES Model" by Cigolini, Pero, Rossi, and Sianesi took a look at some of the phenomena resulting from inherent dependencies/relationships between the performance of a supply chain to the various management decisions and configurations used within the various components of a supply chain (Cigolini, Pero, Rossi, & Sianesi, 2014). They define performance as the occurrence (or conversely the lack of occurrence) of stock levels and stock-outs in a given supply chain (Cigolini et al., 2014). They also break down the various stages within a supply chain to either retailers, distributors, and manufacturers (Cigolini et al., 2014). They narrowed the main parameters in supply chains to number of sources, capacity of system, distance between nodes, number of levels of distribution (how many steps between manufacturer and retailer) (Cigolini et al., 2014).

Some of the phenomena found in this work were rather interesting and may be able to shed light on the incorporation of inventory policy as an element of the LAV depot maintenance model. While the authors determined there was no statistically significant effect of performance due to the distance of nodes, they did notice an increasing trend of stocks outs at the retailer the closer they were to the distributers (Cigolini et al., 2014). They believed this was due to the fact that the retailers based their orders on the lead time due to transportation from distributers and therefore tended to place smaller orders, leading to a lower resistance to demand variation (Cigolini et al., 2014).

To combat this, Cigolini et al. (2014) recommended better information sharing between retailers and distributors. On the part of distributors, they noticed a significant effect due to multiple retailers being supplied, attributed to compounded demand variations due to the individual variances of each retailer's demand; this setup resulted in stock-out and backlogs at the distributors (Cigolini et al., 2014). When retailers decided to split sources from multiple manufacturers, it tended to lower the probability of a stock-out; however, once a stock-out took place it tended to be more severe; particularly problematic for items with a seasonal demand (Cigolini et al., 2014). These effects paint some interesting notions as to how issues that may occur at the depot could be resolved with policy decisions.

"Improving the Rigor of DES in Logistics and Supply Chain Research" by Manuj, Mentzer, and Bowers examined the notion of rigor in simulations; rigor in this case meaning the degree to which complex simulations testing a logistics or supply chain system adhere to a certain set of prescribed standards (Manuj, Mentzer, & Bowers, 2009). The purpose of this paper was to propose their eight step simulation model development process for the design, implementation, and evaluation of logistics and supply chain models (Manuj et al., 2009). The eight steps are (Manuj et al., p. 176):

> "1. Formulate the problem—precisely determining the purpose of the model while retrieving input from all involved stakeholders.
>
> 2. Specify independent and dependent variables.
>
> 3. Develop conceptual model—a walk-through of the process with experts who know it well.
>
> 4. Collect Data.
>
> 5. Develop and verify computer-based model. Based on a detailed flowchart, involve independent programmers, and cross-check against manual calculations.
>
> 6. Validate the model—involve subject matter experts, conduct pilot tests and determine validity of outputs.

7. Run simulations—after appropriately determining proper sample size, runs, and length of each run.

8. Analyze and document results."

These eight steps serve as a fairly apt guide to the creation of any model in the supply chain realm. Certainly, the ultimate model for this LAV depot-level maintenance study will apply these on a case-by-case basis; but as an overarching guideline it is fairly sound.

### 3. Parallel DES

"Parallel Discrete-Event Simulation," by Richard M. Fujimoto delves into the aspect of parallel discrete-event simulation (DES), a method by which we test a discrete-event model by employing parallel computing, that is the use of multiple computer processors to simultaneously work through the computational processes necessary to complete the steps of a given simulation (Fujimoto, 1990).

Parallel DES is in some cases helpful, and in other cases necessary due to the complexity of emerging models (Fujimoto, 1990). The employment of the parallel computing aspect can drastically cut down the time it takes to complete a prescribed number of runs in a given simulation, as there is a limit to how much processing power we can hope to get out of any one processor (Fujimoto, 1990). The concept of parallel DES can often involve an incredible amount of memory and effective networking to ensure that state variables are be accurately updated to reflect the changes made by one processor or another (Fujimoto, 1990). The problems that can arise from parallel DES are that you can have an issue of processing the wrong steps from the event list as you have two separate processers computing events which may at times be dependent on one another (Fujimoto, 1990).

What makes the refinement of parallel DES difficult is that you are in essence attempting to balance the benefit of taking advantage of the additional computing power with the extra memory space, coding, networking capability, and other features you will need to implement in order to get the most efficient result (Fujimoto, 1990). There are essentially two schools of thought when it comes to the implementation of parallel DES, the conservative approach and the optimistic approach (Fujimoto, 1990).

The conservative approach looks to ensure that there are no possibilities that any steps occur out of sequence (Fujimoto, 1990). The program will have code that prevents and event from being processed before the completion of its previously sequenced event (Fujimoto, 1990). While this method avoids the pain of errors in processing order, it can often be overly pessimistic (Fujimoto, 1990), which is in essence to say that it will sacrifice the advantage of computing capability in order to have confidence in the correct processing order (Fujimoto, 1990). The conservative approach will often not take the fullest advantage of parallel computing and will fail to simultaneously process multiple events that may have no dependency on one another (Fujimoto, 1990).

The optimistic approach looks to take the fullest advantage of parallel computing and addresses the errors that may result on the back end by employing measures that detect and recover program errors (Fujimoto, 1990). While the optimistic approach can often take advantage of more simultaneous computing than the conservative approach, the concern is that it does not necessarily mean it is operating more efficiently (Fujimoto, 1990). If too many errors abound, then the concern is that the processors end up spending too much time recovering the correct state of the system instead of primarily engaging the event list effectively (Fujimoto, 1990).

While it is not expected that parallel DES will be required for the depot maintenance model, it is beneficial to understand the challenge and implications it has for the domain of DES.

## C. OTHER SIMULATION APPLICATIONS FOR SUPPLY CHAINS

In “Simulation and optimization of supply chains: Alternative or complementary approaches?” by Almeder et al., the authors specifically mention the need to iterate the testing of the simulation model and the optimization model in order to properly verify and validate the inputs and outputs (Almeder, Preusser, & Hartl, 2009). This is perhaps the biggest takeaway we have from this paper and will come into play much more during the later milestones of the model development.

Once the maintenance depot model has been completed, it will not be sufficient to create parameter adjustments to the system based on baseline performance. We will have to conduct reruns of the optimization as we adjust system parameters in order to confirm which resource parameters levels will ultimately perform at the most effective and efficient levels. “Resources” for purposes of this study are to mean component items, personnel staffing, and maintenance bay space necessary to conduct work; all of which are key ingredients necessary to accomplishing tasks in the system.

In “A simulation-based optimization framework for parameter optimization of supply-chain networks” by Mele et al., the authors also looked at how to best integrate optimization within a discrete-event model (Mele, Guillén, Espuña, & Puigjaner, 2006). The specifics of the analyses do not particularly translate into what will be achieved/attempted with this maintenance depot project, but they do provide some relevant background/proof of concept for the employment of optimization and simulation techniques in tandem to create an effective model.

“Supply chain analysis methodology—Leveraging optimization and simulation software,” by S. Kumar and D. A. Nottestad dealt with the integration of discrete-event simulation and optimization in more industrial sectors (Kumar & Nottestad, 2013). It also discussed how they used the relationship from supplier to distributer and the relationship between inventory control and customer service and how they affect and relate to supply chain policy decisions (Kumar & Nottestad, 2013).

While this particular article was not incredibly relevant in its application of the principles of discrete-event simulation, it did provide very good insights on how to incorporate discrete-event simulation in defining and refining policy decisions, which is ultimately what will be done with the LAV model of depot level maintenance. In addition, this paper also described employing a cost-benefit tool to allow decision makers to see steady state impact of a supply-chain policy decision (Kumar & Nottestad, 2013). While a graphic user interface will be outside the scope of this particular thesis project, it is certainly an interesting idea, and perhaps something that can be considered later on.

# III. APPROACH

One of the primary problems in developing a model of the LAV depot-level maintenance cycle is the size and complexity of the process. In order to provide some method to this effort, the modelling was broken down into several tasks. The first necessary step would be to correctly identify the scope of the problem. What would be in or out of the scope, and to what level of detail would we drill down into the system. Once we determined the system boundaries, it was time to delineate milestones for the model's development, ranging from a basic single server system all the way to the goal of the final system model. The next step would be to gather sponsor input and develop a pilot model. The pilot model would be presented to the sponsor during a site visit in order to demonstrate what a DES product would look like and how their inputs would be used to generate data farming. The site visit was also the time at which we, together with the sponsor, determined the extent to which the model would represent the system, behaviors we would aim to achieve, and the degree to which tasks and events would be aggregated. Following all sponsor inputs, implementation of the final LAV system model could commence

## A. PROBLEM SCOPE

This study looked at any and all operations that take place within the Albany, GA maintenance depot facility. While ultimately LOGCOM will want to have simulations completed for both maintenance depots, it will only be possible at this point to complete a proof of concept for one maintenance depot. Inside the scope of the simulation model will only be actions taking place at the Albany, GA maintenance facility. The scoped actions will "start" with our entity creation/arrival process when vehicles are simulated as being received by the facility.

The scoped actions will stop once a vehicle is considered fully completed with all required maintenance in the system and is ready to be returned to the fleet. It is important to note that that only includes the declaration that the vehicle is ready to be returned to the operating forces; the process of shipping the vehicle will not be modeled. Other vehicles that undergo maintenance at the Albany, GA facility will not be modeled. Actions taken in the operating forces which could affect the ability of MDMC to conduct depot-level repairs on the LAV will be discussed on an anecdotal basis and included as subject matter for future work; but it will not be included in the model itself.

## B. DELINEATE MILESTONES

The initial cut of problem framing was a very high level overview of the system into a single-server concept: an LAV arrives into the depot maintenance system, repairs begin, and repairs end. From that point, the focus was to add complexity to this basic premise and add queue counters, arrival counters, random arrival generators, and random service time generators. At this point, no code is yet being written for the model; everything is being designed through the use of diagrams and event graphs. This high-level approach helps drive the modelling process so that we ensure we are accounting for the appropriate factors before drilling down deep in the wrong direction.

When modelling a complex system such as the entire depot-level maintenance process of the LAV, it is necessary to break the entirety of the modelling into manageable pieces in order to verify a simple iteration of the model before progressing to a more complex iteration. The following milestones were decided upon for the model progression:

1. Basic entity, arrival process, server classes. This milestone was meant to be an initial big picture overview of the system, using a simple single server class to pass information regarding the entirety of the depot maintenance process.

2. Server node aggregation. This milestone expanded the scope of the single server class and incorporated all steps of the maintenance cycle in an aggregated fashion. The maintenance steps were aggregated by the building they were completed at in order to graduate the model's complexity slightly and not jump directly from a single server model into a 156 server model. When completed, this milestone consisted of an arrival process and 22 servers connected in a row by way of adapters. This milestone was used as the basis for the pilot model presented to key sponsor personnel during the site visit.

3. Server complexity. The site visit tour and associated meetings provided additional information that could be added into the individual servers. At this point in

development, the model used separate server class instances for every single step in the maintenance cycle, 156 in total. Additionally, different types of classes were created to more effectively represent what was taking place in the real system, such as rework delays, wedge delays, entity servers, component servers, disassembly servers, assembly servers, servers with bays, among others. Information gathered during the site visit and follow-on communication with the sponsor helped enumerate the probabilities that a particular item underwent a delay, and what the impact of that delay meant for that particular item's maintenance cycle. Additional refinements to individual server service time/resource capacity constraints were compiled during the milestone also. Details on the individual classes will be discussed in more depth in the "Implementation" section of this chapter.

4. Entity complexity. Roughly concurrent with the server complexity development was the development and improvement of the use of entities in the model. As with any coding model, there was a great degree of circular editing and testing before coming to the final granularity of what properties the model needed to account for with a given entity. In the initial stages, the only entities account for in the model was the LAV end item itself. Through development and testing it became more apparent that separate entities classes would need to be created for the key component tracks that the project was looking to represent. This allowed for the model to fully track the maintenance cycle more realistically as well as effectively capture desired statistics in order to show measure of performance.

5. Statistics. Delay in queue, time in system, queue size, number of arrivals, and number of repairs were the statistical figures that were determined most important for tracking in this study. With these figures captured in the model, we would easily see how maintenance cycle time and system throughput would be effected when adjusting resource capacity parameters. Several iterations were required to properly capture these figures. The first several iterations focused on properly capturing the statistical figures across a single run. Once that was successfully achieved, that was designated as the "inner loop" and then collected across multiple run iterations, or an "outer loop" that allowed the user to see what the steady state long run average of those statistics looked like. The outer loop statistics and long run average concepts were especially important for this depot-level maintenance model as the depot maintenance process is very lengthy. Measuring long-run effects across

a live, real-time study would simply be impractical if not impossible. Using simulation to test the effects of resource decisions really allows MDMC leadership to be more prepared before making such decisions.

6. Optimization. After statistics were effectively collected, the next step is to adjust resources capacities in a manner that provides actionable, realistic recommendations to the sponsor that can begin to help ease the bottleneck trends they have experienced in the past. This particular step looked largely at the disassembly, re-assembly, welding, and other time-consuming tasks to see where additional personnel and/or space might be most effectively assigned. See the results and conclusions chapters for more details on how these issues could be addressed.

7. Refinement. The ultimate value of this product to the sponsor is that it is a living software tool that the sponsor can continue to modify and run at their discretion as new or updated information becomes available about the nature of the system capacity, constraints, capabilities, or anything specific to the entities undergoing maintenance. Given the nature of this study and this product, this milestone is never fully complete. A simple adjustment of the input parameters in the model’s main method allows the user to conduct a new run of the simulation and test the effects of the resource adjustment.

## C. PILOT MODEL

Once a basic high-level understanding was developed for the overview of what the model would need to achieve, it was appropriate to begin to collect information specific to the LAV depot-level maintenance process. The first batch of information provided by the sponsor enumerated the various maintenance steps along which the end item travels during its maintenance production cycle. This allowed for the further mapping of the system and helps provide understanding for the level of complexity necessary to study the system. The next set of key products was a work breakdown structure that outlined the production times for each step in the process of both the end item and all component items as well, to include disassembly, re-assembly, inspections, and wedge delays. Additionally, the products also showed how many personnel were required to complete each step in the maintenance cycle.

The pilot model was an important development for a couple different reasons. It provided an initial near sighted goal to work towards prior to any substantial data collection and it also provided a simplified demonstration to show the sponsor what a DES application might look like during the site visit. This gave an incentive to begin the development of several generic Simkit classes for the model which ultimately made the final model easier develop once the available information was mature enough. Having the pilot model ready at the site visit allowed the sponsor to see what their information was going towards and how it would factor into the model. It also helped them see from my perspective why certain types of information about the system were more valuable than others, why the analyzed variables had been chosen in a particular way, and how the granularity of the model had to be broad enough to encompass the full maintenance cycle, detailed enough to capture to behavior of the servers and entities, and yet not too detailed to where the key drivers the study wanted to analyze were not getting lost.

## D. SITE VISIT

The two-day site visit to the Albany, GA, MDMC facility/LOGCOM headquarters was an invaluable experience that helped in achieving the appropriate orientation of the depot maintenance process. Simply reading or having a question and answer session with some subset of the sponsorship group would not have been nearly sufficient to achieve this level of familiarity and understanding. Day one consisted of a lengthy tour of the maintenance facilities and walkthrough of all steps and locations that the end item and major component items traveled throughout the LAV depot-level maintenance process. Following the tour, there was ample time to conduct a meet and greet of various key personnel involved in the depot-level maintenance of the LAV both at LOGCOM headquarters and at the floor of the MDMC depot facility. Day two consisted of various meetings that facilitated more detailed information collection of the LAV process so that the model could more realistically and accurately represent what was going on in the real process. Day two also allowed for a sit in of a teleconference with a Penn State team that is working on a similar project for MDMC, and who very well may attempt to tackle some of the future work recommendations that came out of this study.

## E. MODEL INTENT AND DIRECTION

Once the site visits were completed and the necessary information was acquired, the direction and intent of the eventual model was decided. It was at this point that the decision was made to instantiate every server that has a role in the maintenance of the end item as well as all servers for the miscellaneous hull component, power pack (engine/transmission assembly), suspension system, and communications equipment. In talking with the subject matter experts at MDMC, these four components were the most critical, most problematic, and the most actionable components that could be effectively included into the model (B. Bagley and K. Luckie, personal communication, 30 January 2018).

Employees would be included as a resource parameter at every server instance along with a randomized service time generator per their estimated average real service time. In the assembly, disassembly, and welding server instances, maintenance bay spaces would also be included as a resource parameter. The first goal in building the final model would be correctly tying in a main class all the server instances that reflected the proper flow, probabilities, decisions, delays, etc. that can be observed in the real system.

A guiding feature to identifying intermittent success of this initial piece would be seeing the same issues found in the real system beginning to present themselves in the simulation, such as a sever bottleneck in the welding station. Once the baseline model had achieved some level of fidelity and confidence through numerous simulation runs, we would identify the most problematic servers and adjust the input parameters in order to provide actionable recommendations to MDMC for improving the system's long-run cycle-time and throughput. The key to the previous objective is actionable recommendations; a strict optimization will not be utilized to address this problem as there are simply too many constraints and too little additional resources to work with. The outcome of this particular study will be an output provided to the sponsor showing uses cases in which a slight adjustment of parameters, whether in additional resources or a reallocation of resources, will provide a statistically significant decrease in average cycle time.

Another key thing to remember is that this model will not necessarily provide value in the empirical sense, but in the relative sense. The objective is not to recreate the performance time observed in the real system, nor to achieve some arbitrary level but to demonstrate the degree to which a policy decision improves that performance. Thus, the key performance measure will be the statistical improvement of cycle-time. Further analysis of parametric policy decisions beyond the uses cases provided in the model output results will be considered in future work. For the specific parametric inputs of both the baseline runs and the recommended improvement runs, and their respective outputs, please see the implementation and results chapters.

THIS PAGE INTENTIONALLY LEFT BLANK

# IV. DEVELOPMENT

In creating the model for the LAV depot level maintenance system, as stated previously, DES methodology was the vehicle chosen to represent this system. This chapter will discuss the details of how the DES principles were applied to the LAV depot level maintenance process. This involves a brief background of model theory, DES methodologies, the Simkit software application used to implement the model. The specific classes of code and their behaviors that were developed will be discussed in detail, as will the specifics on how the classes were instantiated and tied together to form the model.

## A. OVERVIEW

In creating a discrete-event simulation model of any system, we must first break down what it means to have a model, a simulation, and that it be discrete-event in nature. A model is often described as an abstraction of reality. In the words of the Panel on Modelling Human Behavior and Command Decision Making (National Research Council, Education, Integration, & Simulations, 1998) “Models are condensed summaries of a domain, omitting (i.e., averaging over) details below a certain level” (p. 186) and that the term “model” itself “implies that human or organizational behavior can be represented by computational formulas, programs, or simulations” (p. 11). This is to say simply that any model is essentially a simplified representation of a given system in order to better understand the nature of the real system (National Research Council et al., 1998). The aforementioned panel goes on to describe a simulation as “a method, usually involving hardware and software, for implementing a model to play out the represented behavior over time” (p. 11).

### 1. Discrete Event Simulation

As mentioned previously, discrete-event simulation (DES) was the vehicle chosen to model the MDMC Albany LAV maintenance cycle. DES is a division under the broader discipline of simulation which is defined mostly in the means by which time is advanced in the simulation (Buss, 2017). In a DES program, time is advanced by irregular steps that are determined by the next event on the event list (Buss, 2017). This is in opposition to a time-step style of simulation which progresses time forward at regular, uniform intervals (Power World Corporation, 2014). A depicting of the next event algorithm is featured in Figure 1.

Whether to implement a model in a time-step or DES method is entirely dependent on which flavor of simulation the analyst believes will address the problem. When the system is being analyzed on the basis of its aggregate performance in continuous procedures over time, a time-step implementation may be appropriate (Power World Corporation, 2014). However, when the analyst needs to address a system that is measuring events taking place, it is imperative that each event be captured discretely for it will be in capturing those events that the holistic value of the simulation will be revealed (Buss, 2017). Additionally, if a time-step simulation method is chosen inappropriately for an event-based model, it could lead to the simulation not detecting the occurrence of one or more events, or not detecting the occurrence at the proper time (Buss, 2017). The effects of this disparity become more apparent when understanding the concept of states, state variables, and events.

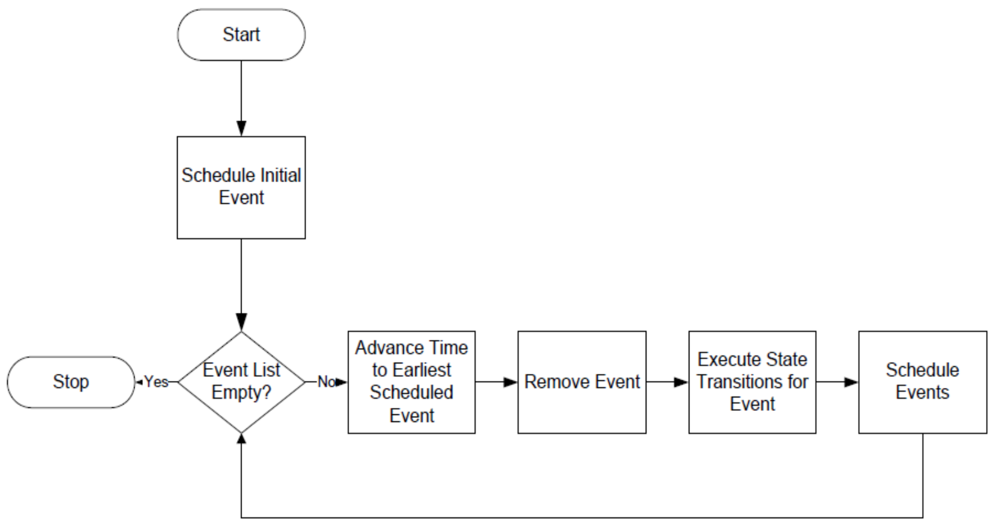


Figure 1. Next Event Selection Algorithm. Source: Buss (2017).

### 2. States and State Variables

The state of a simulation is often described by the value of the changing variables, also called state variables, at any given time in the system (Buss, 2017). State variables are essentially the variables in the model which can be expected to change at different points in time during the simulation's run, such as the size of a queue, the delay in queue of a particular workstation, or the time in system of a vehicle's maintenance cycle (Buss, 2017). State trajectory graphs can be used to measure a single state variable's over time (Buss, 2017). For a DES model, this essentially will be a piecewise function that depicts a state variable's value at varying points in time; the times of which again will be tied to events that are scheduled, added to, and then removed from the event list. Figure 2 features a continuous/time-step state trajectory graph, while Figure 3 depicts the state trajectory graph of a DES system

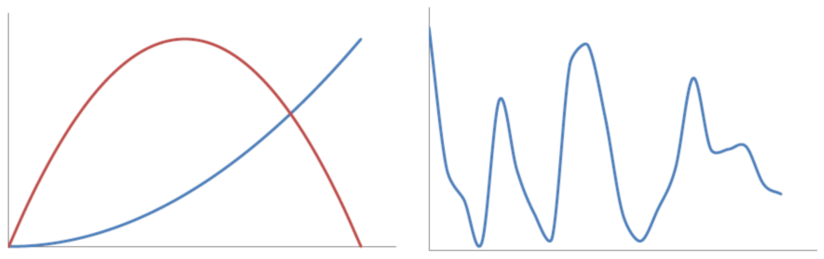


Figure 2. Continuous/Time-Step State Trajectory. Source: Buss (2017).

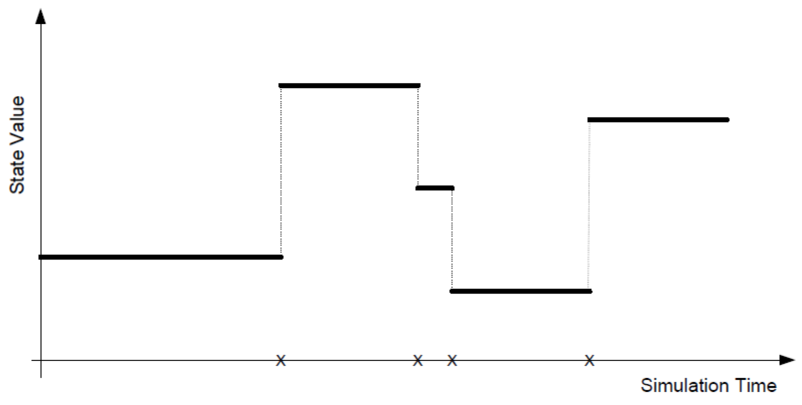


Figure 3. DES State Trajectory. Source: Buss (2017).

### 3. Events

Events are the triggers used in a DES program to employ state transitions (Buss, 2017). Events are called upon to possibly change one, all, or none of the state variables being used in the simulation (Buss, 2017). Events are also used as a means to schedule follow-on events (Buss, 2017). If we consider a pertinent example of a simple maintenance workstation at the MDMC facility, we will have an “Arrival” event that increases the queue size and schedules service if workers are available; a “Start Service” event, which decreases the queue size, decrements the number of available servers, records the delay in queue, and schedules an end of the service task; and an “End Service” event, which records the time in system for that station, increments the number of available servers and schedules another servicing if the queue has items waiting in it (Buss, 2017). In the actual implementation of Java code, these events are represented as methods within a particular class type of the simulation. The object-oriented nature of the java language allows for a relatively seamless application of these events in different class instances (Buss, 2000).

## B. SIMKIT

The tool used to create this simulation model is a Java-based application known as Simkit (Buss, 2000). Simkit was developed by Dr. Arnold Buss at the Naval Postgraduate School as a component based modeling tool by which one can develop generic types of sub-systems, also called “classes,” within a larger system they are attempting to model (Buss, 2000). The user will then create the appropriate number of instances of the types of classes/systems they have designed to represent the model, and connect them together by way of adapters (Buss, 2017). Simkit is essentially a coding capability fulfillment of Dr. Schruben’s Event Graph principle (Buss, 1995). In using Simkit, users are often encouraged to use first create event graphs of their system before developing them in code (Buss, 2017). Because the MDMC Albany facility is composed of numerous maintenance stations that have similar behavior, Simkit was an ideal application with which to develop the model. The Simkit software makes use of various classes, or types of code that will ultimately be implemented in order to represent the appropriate behavior that is found in the system. These classes will be used to represent entities, or objects that are being passed along through the network of service stations; an arrival process, and the service stations that are found in the maintenance process.

## C. CLASSES

### 1. Entities

Entities used this simulation represent the physical LAV in the maintenance cycle as well as key components that are taken from and then re-applied to the end item. Using entity objects as a means by which to map the simulation help us develop a simulation network that makes use of queues, disassembly, reassembly, and other properties and behaviors that help us represent the real system more effectively. Each of the entity classes used in the model are extensions of the Simkit Entity class. The LAV entity class represents the flow of the item through all of its necessary step in the maintenance cycles. The HullMisc class represents floor panels, engine covers, access panels, transmission covers, and other such items disassembled from the hull that must come off and undergo their own sets of procedures. The PowerPack class represents the transmission and engine assembly. The Comm class represents all communications related assets that are pulled off the LAV and repaired/replaced. The Suspension class represents the suspension components travel through its respective maintenance processes.

### 2. LAV Entity Creator

The LAVEntityCreator is a class which extends the Simkit library “ArrivalProcess” class in order to generate instances of the end item entities in the system. The only parameter input for this class is a random variate for the set of interarrival times. Based on the input of the set of random variates, entities are feed into the system and begin working their way through the maintenance cycle. This class is the only instance or extension of an Arrival Process type class found in the model, and it is used to create new *LAV* instances. Component instances are created within the *LAVDisassemblyServer* class. Figure 4 depicts the event graph of LAVEntityCreator class.

*LAVEntityCreator*

Parameters:

{$t_a$}: set of interarrival times

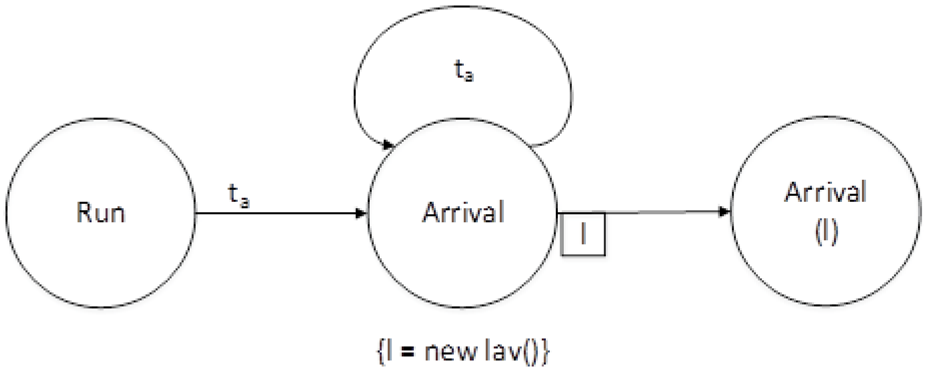


Figure 4. LAVEntityCreator Event Graph

### 3. Entity Servers

Entity servers are the base case instance for any server found in the model. They operate as single service stations with input parameters of a random variate which generates random service times for that particular instance, and the total number of servers. "Servers" in the case of this and all classes is to mean the number of teams of employees that are available to conduct work on a given task. The "Run" method essentially represents the initial instantiation of the class.

When the instance is created, the time in system and delay in queue figures are set to NaN, the available servers is set equal to the total number of servers, and the queue is cleared. When the "Arrival" method is activated, an entity has "arrived" into the system and is added to the queue. The entity is also given a time stamp that starts the delay in queue clock. If there are servers available, then the "Start Service" method is scheduled to take place immediately. Once "Start Service" has been initiated, the entity's delay in queue time, the entity is removed from the queue, and the number of available servers is decreased by one. “End Service” is schedule to take place with a delay time of whatever the generated service time for that particular entity is. “End Service” takes in the entity as an input.

Once “End Service” take places, the entity’s time in system is recorded, and the number of available servers is increased by one. If there are vehicles in the service queue, then another “Start Service” event is also scheduled. The *LAVEntityServer* class is depicted om the event graph in Figure 5, however identical variations of the functionality of these server classes were created that model the HullMisc, Comm, Suspension, and PowerPack servers.

*LAVEntityServer Class*

Parameters

$\{t_s\}$: set of service times
k: total number of servers

State Variables

s: number of available servers
queue: FIFO container of entities
D: delay in queue
W: time in system

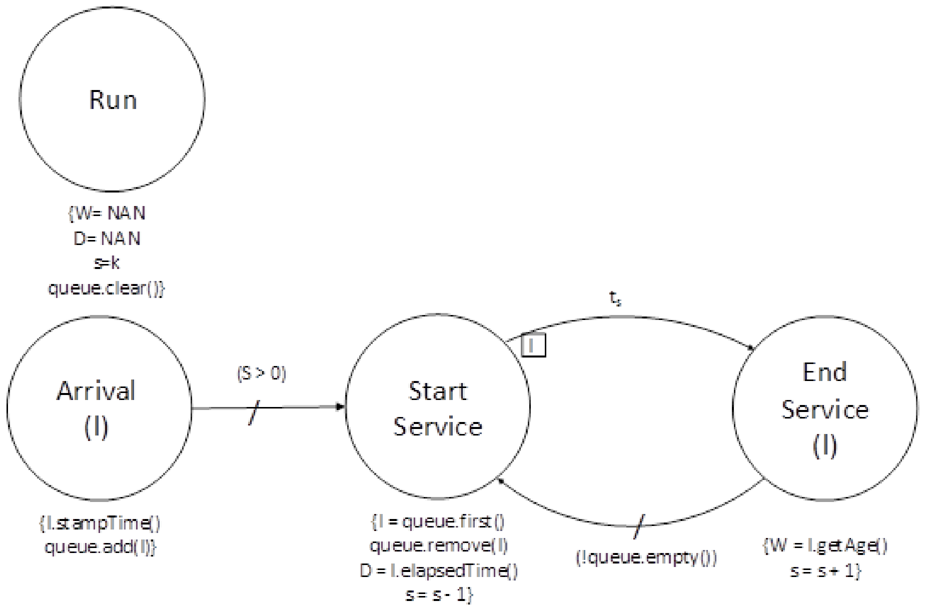


Figure 5. LAVEntityServer Event Graph

### 4. LAV Disassembly Server

The Disassembly server is used in only one place in the model, the station at which the component disassembly is completed and the component items are sent along their respective maintenance routes. This class functions almost identically to the LAVEntityServer; except that upon start service, in addition to the recording of time in system and incrementing the available servers, new component class instances are also created. These component instances are what will be passed to the component entity server networks. A visualization of this class is depicted in Figure 6.

*LAVDisassemblyServer Class*

<u>Parameters</u>

{ts}: set of service times
k: total number of servers

<u>State Variables</u>

s: number of available servers
queue: FIFO container of entities
D: delay in queue
W: time in system

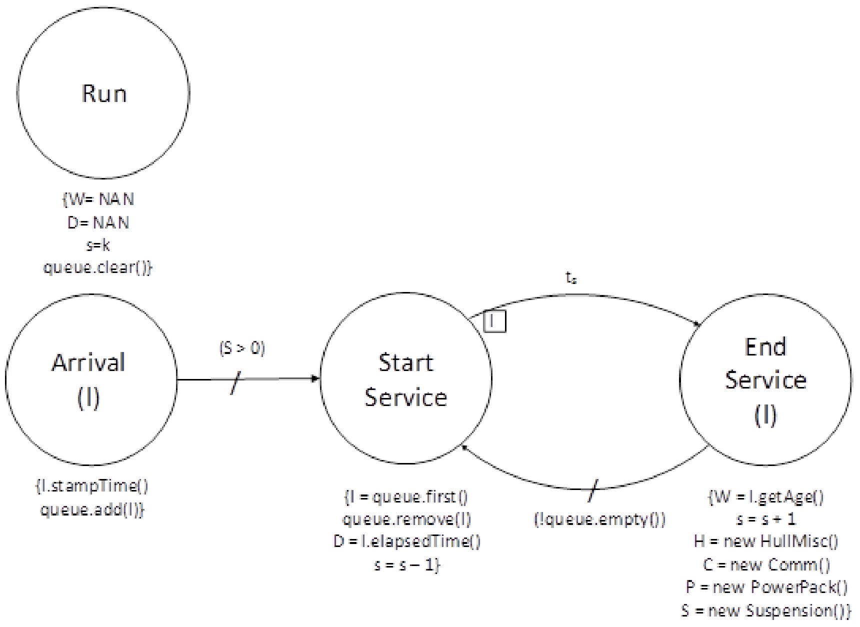


Figure 6. LAVDisassemblyServer Event Graph

### 5. Entity Server with Bays

In modelling the resources used at the facility, it became apparent that it would not be sufficient to use employees alone as a resource. In instances where space was observed to be a concern at the maintenance facility, maintenance bays were also used as a limited resource that must be accounted for when mapping the maintenance cycle. The stations that were represented in the model using this particular class were the disassembly stations and the welding station.

The implementation of this class only required a few slight deviations from the standard entity server classes. An additional parameter is added to the class constructor which defines b: the number of total maintenance bays at that particular workstation. The state variable “a” is then used to identify the number of available maintenance bays at a particular time in the simulation run.

Available maintenance bays are decremented by 1 when the “Start Service” event is activated indicating that the bay is no longer available since a vehicle has gone into maintenance. Once the “End Service” event takes place, the available maintenance bay figures are incremented by 1, representing in the model the notion that the bay is now available to be worked in as a vehicle’s maintenance has just been completed. The event graph of the class is represented in Figure 7.

*LAVEntityServerWithBays Class*

Parameters

{ts}: set of service times
k: total number of servers
b: total number of bays

State Variables

s: number of available servers
a: number of available bays
queue: FIFO container of entities
D: delay in queue
W: time in system

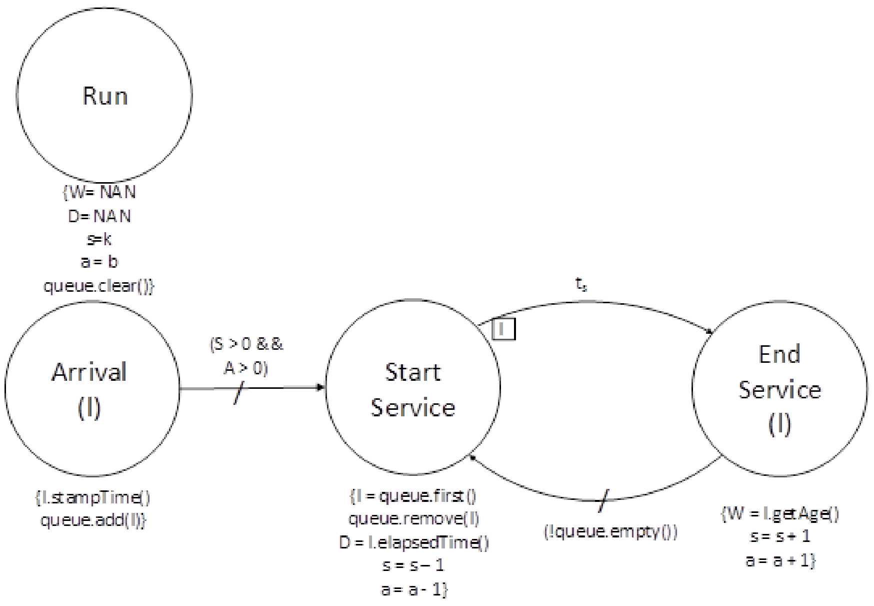


Figure 7. LAVEntityServerWithBays Event Graph

## 6. Wedge Delay Server

In a maintenance facility as complex as the Albany depot, there are several times where items encounter unexpected delays that prolong service time. These instances of unexpected extra time for a particular maintenance step are represented and instantiated by the Wedge Delay Server class. The Wedge Delay server is used primarily for quality control type stations at which spot corrections are made to a vehicle before sending it on its way to complete the remainder of the maintenance cycle.

Examples of stations where the wedge delay server is employed is after an assembly phase in order to ensure everything has been done correctly. As mentioned previously the wedge delay accounts for instances where spot corrections are made to the vehicle in excess of the expected service time of the previous task, and also removal and replacement of a simple subcomponent that may have been flawed with a new or newly refurbished subcomponent from off of the inventory shelf.

New parameters introduced in the wedge delay server class are the delay time and the delay probability. Delay time is the time that a given delay is expected to last in the model. Delay probability is the rate at which we expect a vehicle to fail the inspection/quality control check/etc., and encounter the delay. Upon the "Start Service" event, a Bernoulli generator will generate either "0" or "1" at an average long-run rate determined by the delay probability parameter; this value is stored under the Boolean variable "delay." If "delay" is found to be 0, that indicates that the vehicle passed its inspection and proceeds without delay to schedule an "End Service" event. If the "delay" variable is found to be 1, that indicates that an issue has been detected and a delay will be incurred.

When a delay must take place in the model, the entity is transferred to the "Delay" event, which will then successively schedule the "End Service" event with a wait time of the indicate delay time parameter. As the wedge delay is taking place on the spot, only an entity's activation of the "End Service" event will free up the team of employees working on the vehicle. This key difference is one of the things that distinguishes this delay class from the Rework Delay server class we will examine later on. The event graph for this class is represented in Figure 8.

*WedgeDelayLAVServer Class*

Parameters

{ts}: set of service times
k: total number of servers
dt: delay time (double, constant)
dp: delay probability (double)

State Variables

s: number of available servers
queue: FIFO container of entities
D: delay in queue
W: time in system
bernoulli = random variate generator based on delay probability

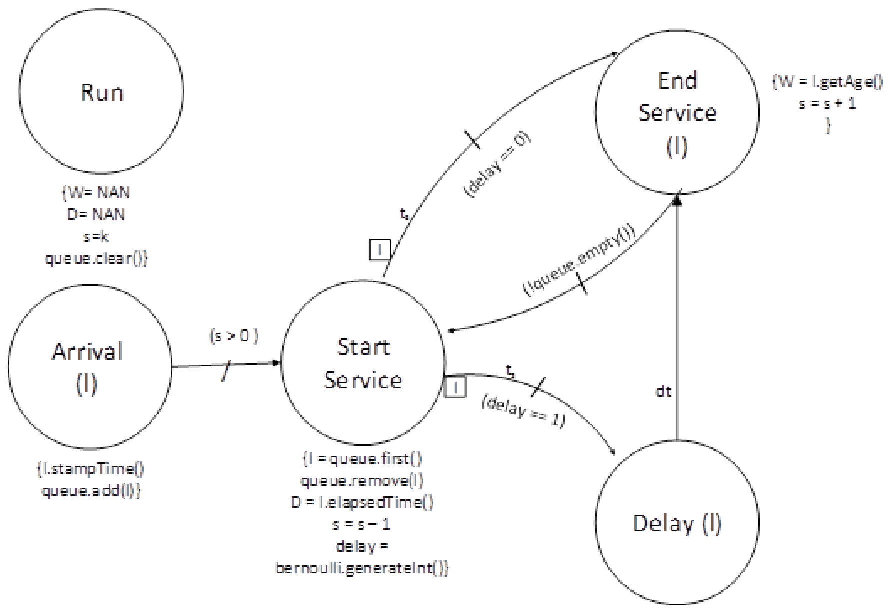


Figure 8. WedgeDelayLAVServer Event Graph

## 7. Rework Delay Server

The Rework Delay server class will have quite a few similarities to the Wedge Delay server class we examined previously. The Bernoulli generator will generate upon the "Start Service" event in the same fashion as before and determine whether an event will "pass" and go on to the "End Service" event or whether an event will "fail" and be sent to the “Delay” event. The key difference in this class is that there is no parameter for delay time. The delay time in this particular class will take effect when it is sent from the “Delay” event of this class instance to the “Arrival” event of the station it is being sent back to for additional work.

The only case the Rework Delay server is being used in this model is the heat distortion temperature (HDT) inspection following the welding station’s entity server. Following the completion of a welding job, the vehicle is sent to receive an HDT inspection in order to ensure that the hull integrity meets specifications following completion of the welding job. If the vehicle passes, it proceeds further into the maintenance cycle. If the vehicle fails, it is sent back through the welding station. In order to effectively model this class, a variable was created for the LAV class named “numberWeld,” which indicates the amount of times the vehicle has been through the welding station. Initially set the 0, numberWeld is incremented by one for that entity each time it is sent to the “Delay” event in the model.

The model employs two Bernoulli generators in determining how to send vehicles to rework. The first generator causes the vehicles to fail at a rate of .7. The second generator causes vehicles to fail at a rate of .1. The desired effect being that while vehicles are expected to require some rework 70% of the time after the first inspection, only 10% of vehicles are expected to require rework after the second inspection, and all vehicles are expected to pass a third inspection. This behavior is based on subject matter expertise on the failure rates of vehicles going back for additional welding/HDT inspections (K. Luckie, personal communication, 11 April 2018). Because the vehicles leave the station upon activating the “Delay” event, it also changes the states variables similarly to an “End Service” event and frees up a team of employees to work on the next LAV in the queue. Figure 9 depicts the event graph for this class.

*ReworkDelayLAVServer Class*

<u>Parameters</u>

{ts}: set of service times
k: total number of servers
dp: delay probability (double)

<u>State Variables</u>

s: number of available servers
queue: FIFO container of entities
D: delay in queue
W: time in system
bernoulli = random variate generator based on delay probability

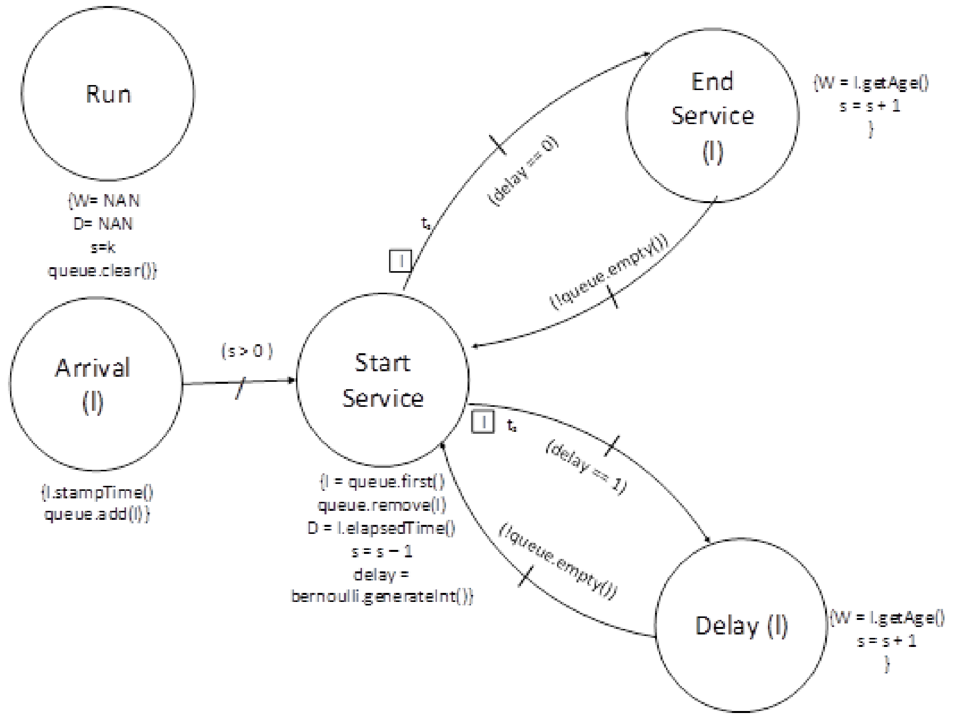


Figure 9. ReworkDelayLAVServer Event Graph

### 8. Component Re-Assembly Servers

The Component Assembly server classes are designed specifically to take in two separate types of entities and simulate the performance of the assembly process and then sending the LAV entity itself along the remainder of its route in the maintenance cycle. The major distinguishing feature with this class is that it has two different arrival events. “LAV Arrival” is where the LAV entity arrives and is added to its respective queue. The “Component Arrival” event adds a component entity to the component queue.

Conditions for proceeding to the “Start Service” event are having entities ready in both the LAV and component queues, space available, and server teams available. Upon “Start Service” the entities are removed from their respective queues, and states variables recorded appropriately. From that point, the class behaves nearly identically to the “LAVEntityServerWithBays” class. The event graph for this class type is depicted in Figure 10.

________*AssemblyServer Class* (Comm, HullMisc, PowerPack, and Suspension)

Parameters

{ts}: set of service times
k: total number of servers
b: total number of bays

State Variables

s: number of available servers
a: number of available bays
queuel: FIFO container of LAV end-item entities
queueh: FIFO container of component entities
D: delay in queue
W: time in system

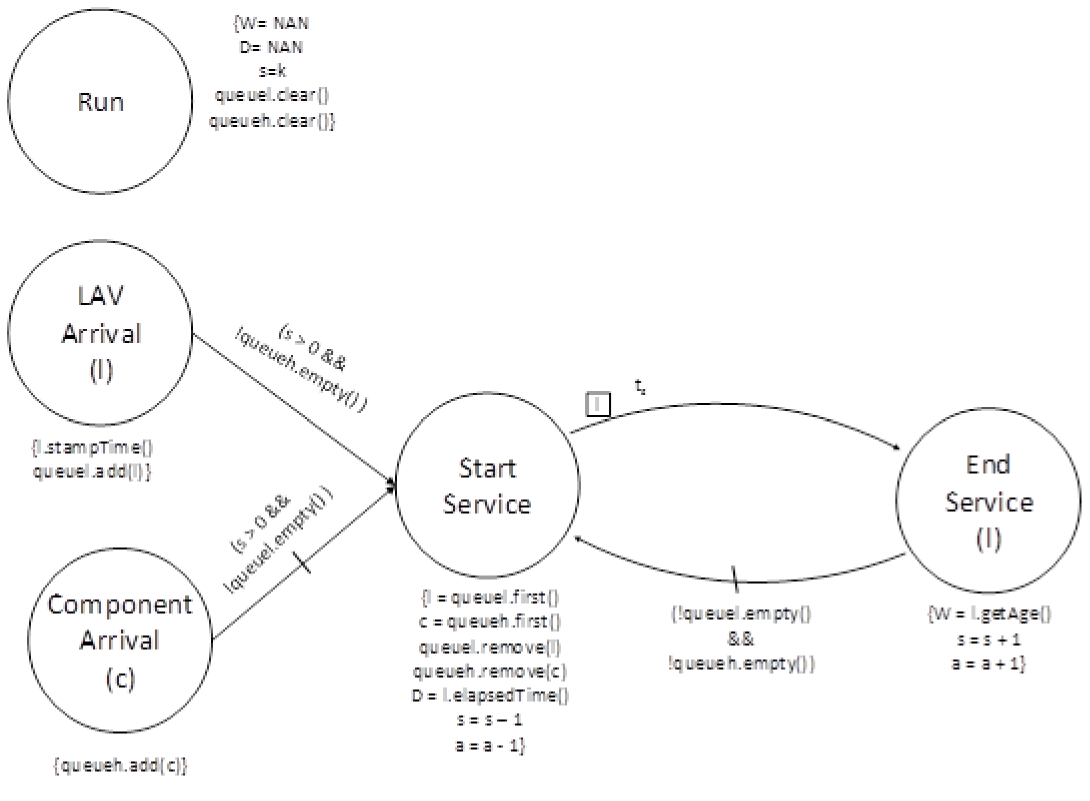


Figure 10. Component Assembly Server Event Graph

## D. IMPLEMENTATION

In creating the main method for the simulation, we faced three major challenges: compiling the appropriate data to create random number generators for the arrival times and service times for each class instance, instantiating the appropriate classes in the model as necessary, and connecting those classes together appropriately such that they represent the behaviors observed in the LAV depot-level maintenance process.

An exponential distribution was used to create random interarrival times of the LAVs into the system. The system averages about 30 LAV arrivals/year and the exponential distribution was used to provide some variability in that effect on the front end of the maintenance process. Unfortunately, there was no data on service times available with which to apply a distribution for the model; we only had the average service times for the maintenance steps being analyzed. In order to achieve some variability in service times, the service times were all applied with a gamma distribution and a shape factor of 2. After consultation with the advisor for the study, we both concurred that these distributions would be acceptable to achieve the behavior effects the model was designed to represent (A. Buss, personal communication, 7 February 2018). However, the simulation model is flexible so that if subsequent data indicate different service time distributions, they could be seamlessly incorporated into the model. Once all random number generators are created, the server classes are all instantiated as appropriate. The respective random number generator that pertains to a given class is input as a parameter as well as the other parametric inputs. Based on inputs from the depot, baseline personnel policy inputs were entered as well.

Once the class instances were all created, we then create the appropriate listeners and adapters that will be used to connect the classes they are passing entities between. Only one listener was instantiated in the system which was used to connect the *LAVEntityCreator* class to the first server class instance in the system. The adapter instantiations only have two inputs—the event that the adapter is accepting an entity from, and the event that the entity is being passed to. The adapter instances are independent of the servers they are connecting; only once we connect the servers with the method “adapter1.connect(server1, server2)” are the servers in the model fully connected. The connection of one class instance to another can be depicted using a listener diagram (Buss, 2017). Figure 11 depicts the listener diagram for the *LAVEntityCreator* class instance connecting to the first *LAVEntityServer* class instance. Figure 12 depicts the listener diagram of the *LAVDisassemblyServer* class instance to the first entity server class instances of each of the major components that were analyzed. Figure 13 depicts the listener diagram of a component assembly server class instance taking inputs from both the preceding *LAVEntityServer* instance and the final component entity server of the component being assembled.

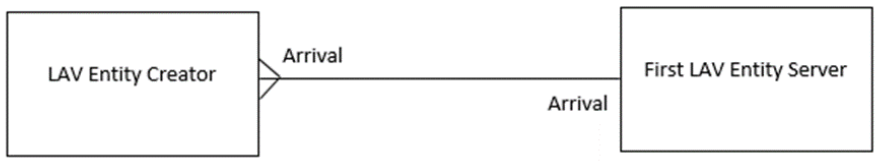


Figure 11. Listener Diagram Depicting Entities being Passed from the LAV Entity Creator Class to the First LAV Entity Server Instance

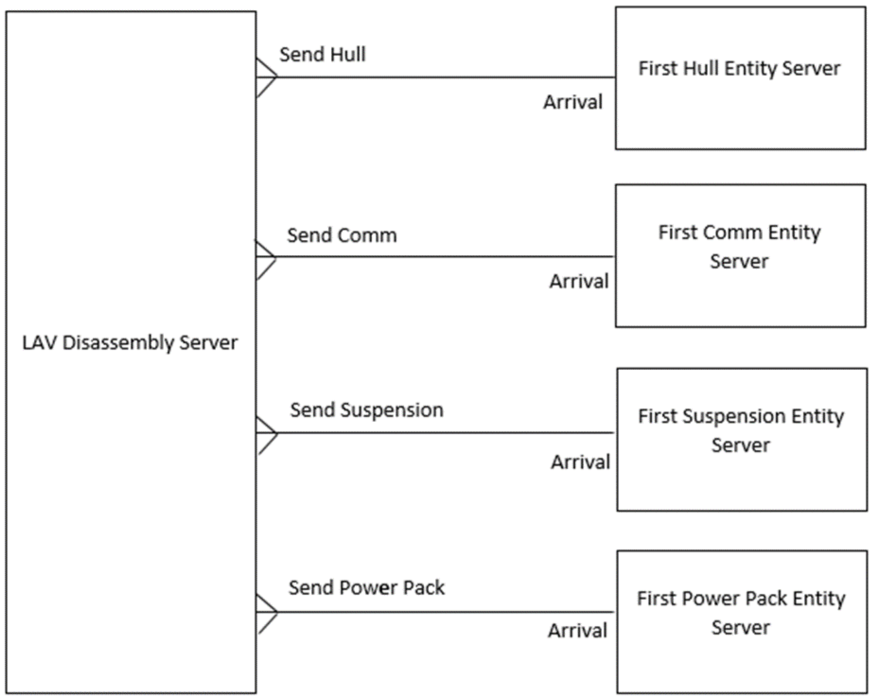


Figure 12. Listener Diagram Depicting Component Entities being Created by the LAV Disassembly Server Instance and Passed to their Respective Entity Server Networks

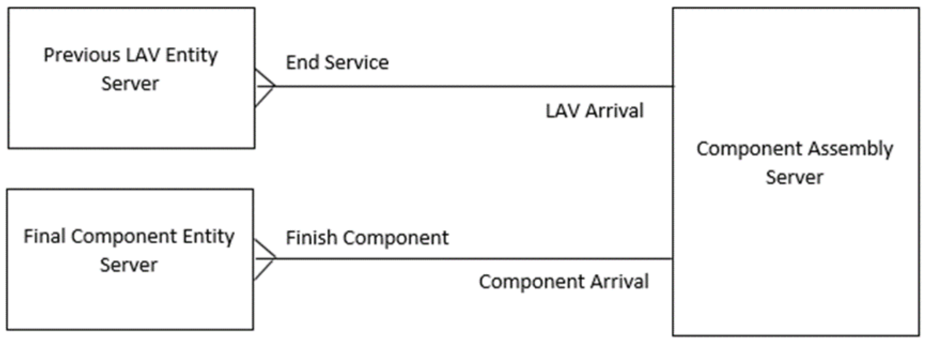

Previous LAV Entity Server
End Service
LAV Arrival
Component Assembly Server
Final Component Entity Server
Finish Component
Component Arrival

Figure 13. Listener Diagram Depicting the LAV and Component Entities being Passed to the Respective Component Assembly Server Instance

Next, the loggers for the model are instantiated to collect data from the simulation runs. Multi-tally statistics loggers are used for the "delay in queue" and "time in system" statistics. The "queue" statistic is collected with a multi-collection size data logger. Each of these loggers collects the statistics at appropriate points in the model in order to capture the information that is being studied, compile it correctly, and write to a .csv output file. The "delay in queue" and "queue" statistics capture every single queue in the system, while the "time in system" statistic captures only one value, the time in system statistics at the last maintenance station in the system. Property change listeners are added to the code in order to the loggers are capturing the necessary information from each server. The property change listeners also ensure the same statistical information is captured and output to the console also, as this helps in debugging.

Following the property change listeners, an iteration loop is added which contains the lines of code that actually run and govern the simulation time parameters. The loop resets all servers in the system to initial conditions and runs them for roughly a five-year time period. While Simkit is operating on a pseudo-random generation which is technically deterministic, the random number generator used is known to have excellent properties in mimicking randomness. Furthermore, the iteration loop allows the servers to be reset and multiple identical independent replications be performed. This is crucial for estimating the performance measures with statistical accuracy.

The last bit of code in the main method running class contains the print statements. These statements largely contain statistical information that can be quickly accessed on the console and helped largely in the debugging efforts of the simulation model's development and refinement.

# V. RESULTS

The testing of the LAV depot-level maintenance model was conducted in three stages. The first stage was a baseline run that was designed to determine which maintenance steps in the process had a delay-in-queue that was significantly larger than the rest of the system. The next stage consisted of an iterative improvement of the system—incrementing the resources of the problematic servers until no server in the system had a delay-in-queue statistic that was significantly larger than the rest of the system. This was meant to understand how other servers in the system might behave as bottlenecks were addressed in one part of the system or another. The final stage of testing was to conduct experiments combining different arrangements of parameter inputs for the problematic service stations in order to determine which servers had the most statistical impact on the ultimate goal of reducing the service time in the system.

## A. MODEL GOVERNANCE

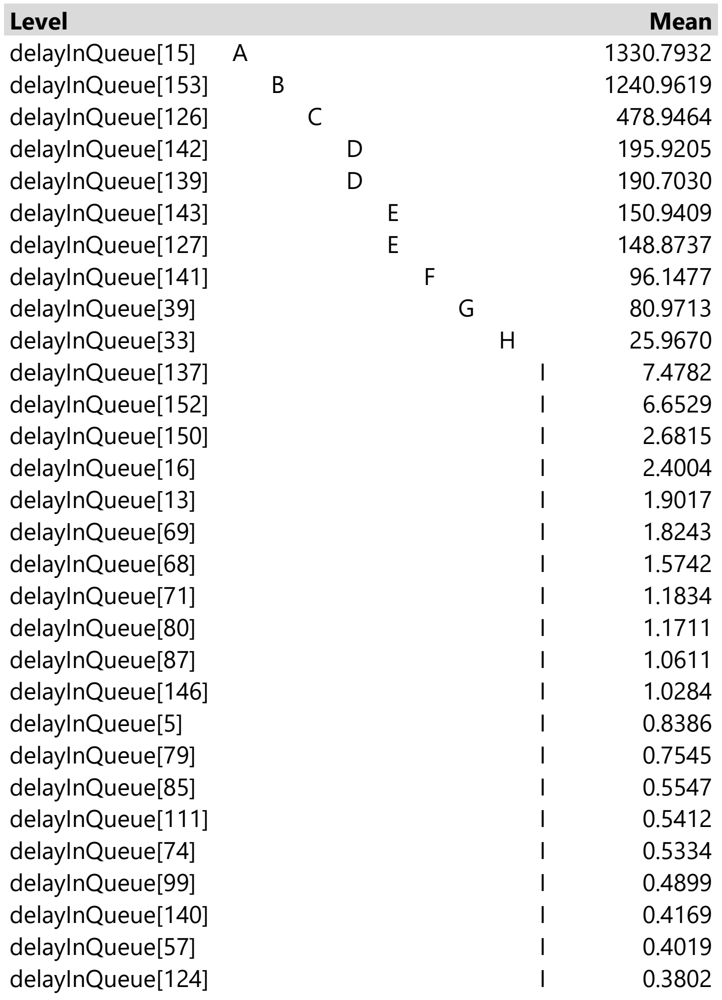
Each replication in the running class ran the model for a five-year time period. In order to obtain a high confidence in the long-term performance of a given set of input parameters for the model, each simulation run consisted of 1000 independent replications. Following completion of the simulations run, csv files were written containing data for the queue size changes, average delay in queue per server per replication, and time in system per replication.

The delay in queue statistic was the measure used to determine which server instances needed to have their resource capacities expanded to alleviate the bottlenecking that was taking place. The delay-in-queue statistic is the best indicator to determine the amount of time in a given entities maintenance cycle that is lost due to waiting at that particular queue. The csv file for a particular run's delay-in-queue statistic was imported to JMP statistical analysis software (JMP Pro Version 13.1.0., 2016) and ANOVA tests performed in order to determine which servers had delay in queue statistics were significantly above the average for the entire system; in particular, the connecting letters report within the Tukey-Kramer HSD test (JMP Pro Version 13.1.0., 2016). Those servers then had their resource capacities incremented by 1 (server teams and bays, if applicable). This improvement process was iterated until there was no longer a significant difference present between successive servers when ordered from longest to shortest average queue delay. Theoretically this could result in an effect of regression to a long average delay between servers; however, that was not a concern as most servers even after the initial baseline run had very short queue delays, at or near zero.

Following the completion of all iterative improvement steps and follow-on runs of the simulation model, the time in system statistics were used to measure the improvement of the cycle time in the system across multiple runs.

## B. BASELINE RUN

The baseline run of the model was intended to represent the system’s long-run performance based on current resource parameters. Upon completion of the baseline run, the delay-in-queue output for all servers was opened in JMP statistical analysis software and was tested with an ANOVA tests as well as a means comparison/Tukey’s honest significant difference (HSD) test. The output from the connecting letters report as part of these tests arranges all servers in order from the most extreme to least extreme average delay-in-queue time. The output shown in Figure 14 is an excerpt from the ANOVA Tukey’s HSD connecting letters report, and was the primary device used to determine which servers in the system were targets for improvement following any given simulation run.



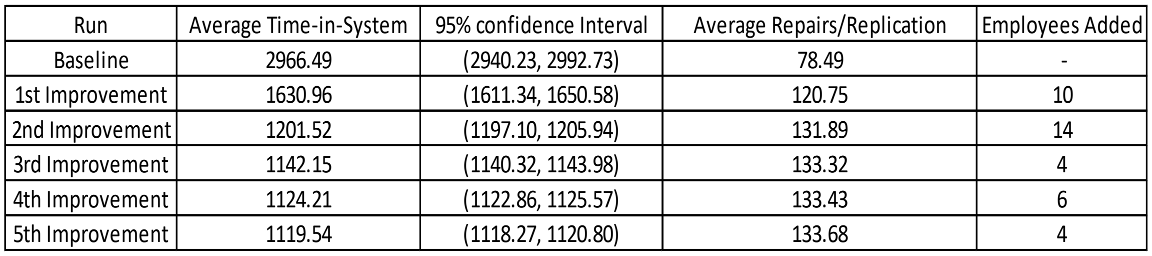
| Level | | Mean |
|---|---|---|
| delayInQueue[15] | A | 1330.7932 |
| delayInQueue[153] | B | 1240.9619 |
| delayInQueue[126] | C | 478.9464 |
| delayInQueue[142] | D | 195.9205 |
| delayInQueue[139] | D | 190.7030 |
| delayInQueue[143] | E | 150.9409 |
| delayInQueue[127] | E | 148.8737 |
| delayInQueue[141] | F | 96.1477 |
| delayInQueue[39] | G | 80.9713 |
| delayInQueue[33] | H | 25.9670 |
| delayInQueue[137] | I | 7.4782 |
| delayInQueue[152] | I | 6.6529 |
| delayInQueue[150] | I | 2.6815 |
| delayInQueue[16] | I | 2.4004 |
| delayInQueue[13] | I | 1.9017 |
| delayInQueue[69] | I | 1.8243 |
| delayInQueue[68] | I | 1.5742 |
| delayInQueue[71] | I | 1.1834 |
| delayInQueue[80] | I | 1.1711 |
| delayInQueue[87] | I | 1.0611 |
| delayInQueue[146] | I | 1.0284 |
| delayInQueue[5] | I | 0.8386 |
| delayInQueue[79] | I | 0.7545 |
| delayInQueue[85] | I | 0.5547 |
| delayInQueue[111] | I | 0.5412 |
| delayInQueue[74] | I | 0.5334 |
| delayInQueue[99] | I | 0.4899 |
| delayInQueue[140] | I | 0.4169 |
| delayInQueue[57] | I | 0.4019 |
| delayInQueue[124] | I | 0.3802 |

Figure 14. Baseline Delay-in-Queue: Connecting Letters Report Excerpt. Source: JMP® Pro Version 13.1.0 (2016).

## C. ITERATIVE IMPROVEMENT

The baseline run ANOVA/Tukey's HSD connecting letters report output displayed the comparison of the average delay-in-queue statistics from each server in the system. The connecting letters report tells us that servers that are not connected by the same letter have a significant difference in average delay-in-queue time. Because the model generated the statistics of each server from a base zero frame of reference, we know that servers 16, 154, 127, 143, 140, 144, 128, 142, 40, and 34 have significantly longer average delays in their queues. These servers had their resources incremented by 1 prior to the start of the next run of the simulation.

This process of delay-in-queue analysis, incrementing the server resources, and running the model with adjusted parameters constituted the first improvement iteration step. In total, five improvement iterations were conducted for this model. Conditions to stop the iterative improvement process was the lack of significant difference between server delay-in-queue statistics. The effects of the improvement steps were measured by comparing the time in system statistics following each run. The table featured in Figure 15 was assembled by consolidating the average time-in-system for each run iteration as well as a 95% confidence interval for time-in-system, throughput measured in average number of repairs/replication for that run, and the number of employee teams that were added to the system for that particular run of the simulation.



| Run | Average Time-in-System | 95% confidence Interval | Average Repairs/Replication | Employees Added |
|---|---|---|---|---|
| Baseline | 2966.49 | (2940.23, 2992.73) | 78.49 | - |
| 1st Improvement | 1630.96 | (1611.34, 1650.58) | 120.75 | 10 |
| 2nd Improvement | 1201.52 | (1197.10, 1205.94) | 131.89 | 14 |
| 3rd Improvement | 1142.15 | (1140.32, 1143.98) | 133.32 | 4 |
| 4th Improvement | 1124.21 | (1122.86, 1125.57) | 133.43 | 6 |
| 5th Improvement | 1119.54 | (1118.27, 1120.80) | 133.68 | 4 |

Figure 15. Improvement Step Comparisons. Source: JMP Pro Version 13.1.0. (2016).

The results here demonstrate that each improvement step provided a significant reduction in overall system cycle-time. This is given from the fact that the average time-in-system confidence interval's upper bound for an improvement step in the simulation is always lower that the lower bound of the preceding run's confidence interval for its average time-in-system.

## D. REGRESSION

The iterative improvement step was only the first piece of the testing and analysis that was conducted with this model. Once all improvement steps were completed, each server that had undergone improvement was arranged into a 16-factor nearly orthogonal

Latin hypercube (NOLH) experimental design array. This array generator tool was provided by the Naval Postgraduate Schools Simulation, Experiments, and Efficient Design (SEED) Center for Data Farming (“Naval Postgraduate School SEED Center website”). The 16-factor NOLH array was generated from a spreadsheet tool which took as inputs the “low level” and “high level” of each server (“Naval Postgraduate School SEED Center website,”). The low level is essentially the baseline resource parameter setting for a particular server. The high level is the maximum number of resources a given service station had across all improvement steps in the simulation.

From these inputs, the spreadsheet displays the appropriate resource parameters that each of the 16 altered servers needs to be set at across an additional 65 runs of the simulation. This testing method, while it appears to be time-consuming, it actually saves the user time because it allows us to detect behavior of the system at different parametric settings not observed during the improvement steps and explore the space of those parametric possibilities without testing every single parameter in the space. Following the completion of each of the runs, the average time-in-system across all replications within that run was recorded.

When all runs were completed, the server parameters and time-in-system performance for each run was put into JMP in order to test the effects of server parameter settings on system performance. In order to achieve that goal, the data was run through a multiple linear regression in JMP with the explanatory variables being the various server resource parameter settings for a run and the outcome variable being the system’s time-in-system for that run. Ultimately, the objective with this test was to observe the significance to which each server’s resource parameters affected the time in system of the model. The below output shows the key finding from the multiple regression. We can see from these outputs that the multiple linear regression fit of actual by predicted plot (Figure 16) that the model demonstrates a fair correlation with an R-Square value of .49. We can also see from the output of the effects summary (Figure 17), effect test (Figure 18), and parameter estimates (Figure 19) that there is a statistically significant effect of servers 16 and 113F on the model.

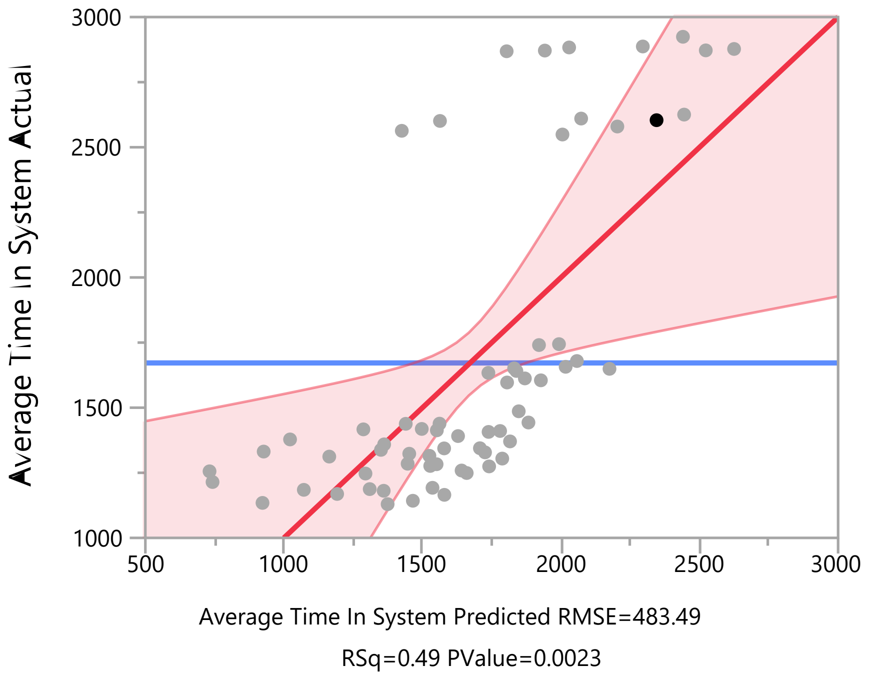
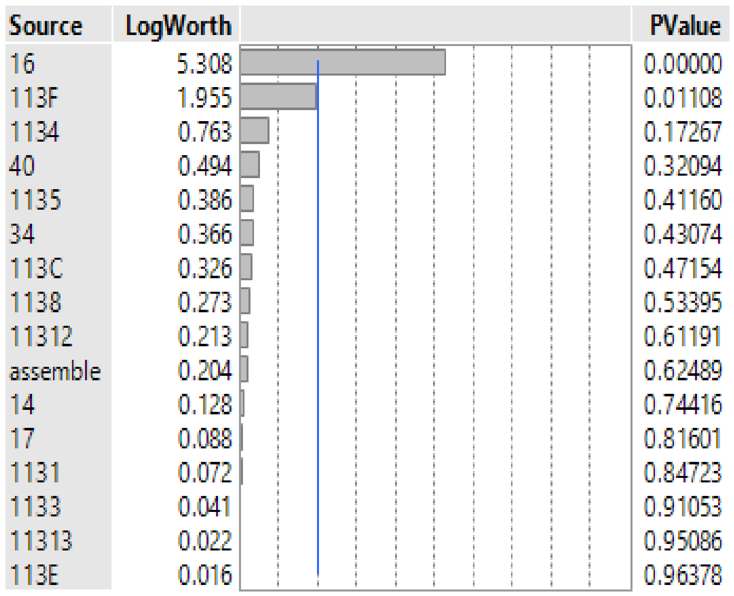
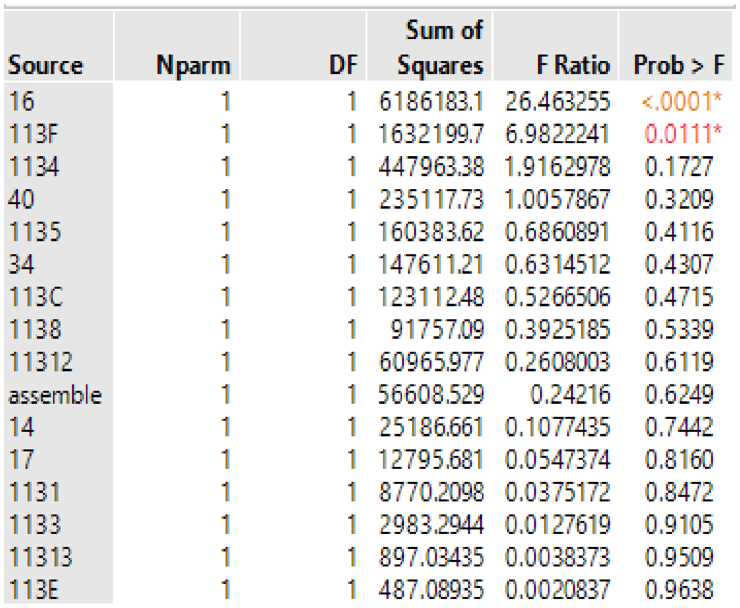


Figure 16. Actual by Predicted Plot. Source: JMP Pro Version 13.1.0. (2016).



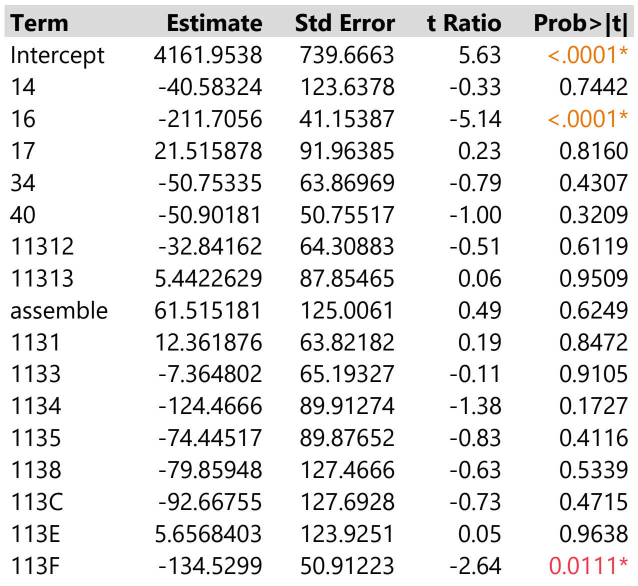
| Source | LogWorth | PValue |
|---|---|---|
| 16 | 5.308 | 0.00000 |
| 113F | 1.955 | 0.01108 |
| 1134 | 0.763 | 0.17267 |
| 40 | 0.494 | 0.32094 |
| 1135 | 0.386 | 0.41160 |
| 34 | 0.366 | 0.43074 |
| 113C | 0.326 | 0.47154 |
| 1138 | 0.273 | 0.53395 |
| 11312 | 0.213 | 0.61191 |
| assemble | 0.204 | 0.62489 |
| 14 | 0.128 | 0.74416 |
| 17 | 0.088 | 0.81601 |
| 1131 | 0.072 | 0.84723 |
| 1133 | 0.041 | 0.91053 |
| 11313 | 0.022 | 0.95086 |
| 113E | 0.016 | 0.96378 |

Figure 17. Effect Summary. Source: JM Pro Version 13.1.0. (2016).



| Source | Nparm | DF | Sum of Squares | F Ratio | Prob > F |
|---|---|---|---|---|---|
| 16 | 1 | 1 | 6186183.1 | 26.463255 | <.0001* |
| 113F | 1 | 1 | 1632199.7 | 6.9822241 | 0.0111* |
| 1134 | 1 | 1 | 447963.38 | 1.9162978 | 0.1727 |
| 40 | 1 | 1 | 235117.73 | 1.0057867 | 0.3209 |
| 1135 | 1 | 1 | 160383.62 | 0.6860891 | 0.4116 |
| 34 | 1 | 1 | 147611.21 | 0.6314512 | 0.4307 |
| 113C | 1 | 1 | 123112.48 | 0.5266506 | 0.4715 |
| 1138 | 1 | 1 | 91757.09 | 0.3925185 | 0.5339 |
| 11312 | 1 | 1 | 60965.977 | 0.2608003 | 0.6119 |
| assemble | 1 | 1 | 56608.529 | 0.24216 | 0.6249 |
| 14 | 1 | 1 | 25186.661 | 0.1077435 | 0.7442 |
| 17 | 1 | 1 | 12795.681 | 0.0547374 | 0.8160 |
| 1131 | 1 | 1 | 8770.2098 | 0.0375172 | 0.8472 |
| 1133 | 1 | 1 | 2983.2944 | 0.0127619 | 0.9105 |
| 11313 | 1 | 1 | 897.03435 | 0.0038373 | 0.9509 |
| 113E | 1 | 1 | 487.08935 | 0.0020837 | 0.9638 |

Figure 18. Effect Tests. Source: JMP Pro Version 13.1.0. (2016.)



| Term | Estimate | Std Error | t Ratio | Prob>\|t\| |
|---|---|---|---|---|
| Intercept | 4161.9538 | 739.6663 | 5.63 | <.0001* |
| 14 | -40.58324 | 123.6378 | -0.33 | 0.7442 |
| 16 | -211.7056 | 41.15387 | -5.14 | <.0001* |
| 17 | 21.515878 | 91.96385 | 0.23 | 0.8160 |
| 34 | -50.75335 | 63.86969 | -0.79 | 0.4307 |
| 40 | -50.90181 | 50.75517 | -1.00 | 0.3209 |
| 11312 | -32.84162 | 64.30883 | -0.51 | 0.6119 |
| 11313 | 5.4422629 | 87.85465 | 0.06 | 0.9509 |
| assemble | 61.515181 | 125.0061 | 0.49 | 0.6249 |
| 1131 | 12.361876 | 63.82182 | 0.19 | 0.8472 |
| 1133 | -7.364802 | 65.19327 | -0.11 | 0.9105 |
| 1134 | -124.4666 | 89.91274 | -1.38 | 0.1727 |
| 1135 | -74.44517 | 89.87652 | -0.83 | 0.4116 |
| 1138 | -79.85948 | 127.4666 | -0.63 | 0.5339 |
| 113C | -92.66755 | 127.6928 | -0.73 | 0.4715 |
| 113E | 5.6568403 | 123.9251 | 0.05 | 0.9638 |
| 113F | -134.5299 | 50.91223 | -2.64 | 0.0111* |

Figure 19. Parameter Estimates. Source: JMP Pro Version 13.1.0. (2016).

## E. CONCLUSIONS

From the results of the simulation runs and statistical analysis we have seen that expanding server resource capacity can have a tremendous effect on the time-in-system performance of the system. The improvement step comparisons demonstrated that a significant reduction in repair cycle-time can be achieved by adding additional resources to the slower servers in the maintenance cycle. Not all queues that were experimented with following the improvement step analysis were found to have a statistical impact on the overall time-in-system of the LAV.

From our statistical analysis, it is clear that the system would greatly benefit from capacity expansion at station 16 (welding) and station 113F (driving differential repair, paint, and assembly); the other stations, while having the potential to relieve somewhat larger delay-in-queue times, should not be expected to have a statistical impact on the system’s performance. This is important to understand, because any organization with a limited investment budget, such as MDMC, will have a limited amount of resources that can be devoted to improving any one maintenance process. The NOLH analysis provides the sponsor, in order of precedence, the degree to which resource availability at a given server will provide them a return in reduced maintenance cycle-time, and thus provides a plan of action and justification for allocating additional resources towards a different maintenance station in the system.

# VI. FUTURE WORK

This study was the first of its kind at MDMC; DES had never before been used to study an entire vehicle's maintenance cycle. Through the research and work that was done to create a worthwhile model, several follow-on topics were identified that could be beneficial to the process improvement endeavors at the MDMC and the discovery of knowledge that might help LOGCOM better understand the depot-level maintenance processes of not just the LAV, but many types of equipment throughout the Marine Corps.

As mentioned previously one of the challenges of undertaking this study was the lack of data to work with. The aspect of the model that was most affected by this disparity was the random number generation that was used for the service times of the service stations in the model. In order to have a full verification, validation, and accreditation (VV&A) of this model, the service times used in the model would need to reflect the actual distribution of service times found in the system.

In order to do this, a robust data collection effort would have to be started at MDMC. Currently data is not collected by maintenance step, the data collection effort is focused on the labor hours completed by a particular work section of employees over a specified span of time. As the current cycle-time for the LAV going through depot-level maintenance is estimated around 180 days (K. Luckie, personal communication, 29 January 2018), it would likely take several years to obtain the necessary data required to have a basis to adjust the service time random number generators throughout the model.

As this model was designed through the use of an object-oriented programming application, the classes developed could be instantiated to represent servers and behaviors of any other equipment's maintenance cycle at MDMC. In addition to modelling other types of equipment, a future study could look at increasing the complexity of the existing class to include more elaborate behaviors, more statistical tracking, and an integration of additional business metrics that would be beneficial to the decision makers at MDMC.

Another key thing to keep in mind about the model is that it only looks at analyzing and improving the system performance of steady state behavior. That is to say that this model looks at how the system can be expected to perform on the average over a predetermined period of time. Another study could look at including a flexible schedule/flexible state construct for the model. For example, it might look at when it would make sense to incorporate an overtime policy at MDMC. That simulation could compare how overtime policies are currently managed at MDMC vs alternative and perhaps more efficient methods. Additionally, this study could include a cost-benefit analysis of adding additional resources to servers in a steady-state long term case compared to other cases of adjusting overtime policies on a flexible state basis, or some combination of the two.

Outside the influence of MDMC exists an entire realm of possibilities that could be addressed in a future study. There are a variety of issues outside MDMC that have an impact on maintenance cycle-time and repair costs. One such issue is the fact that MDMC does not operate on the same oracle-based software that the operating forces are using for maintenance management. This results in MDMC not having access to the full complement of property and repair requisition records. This can have the effect of MDMC not having oversight of some of the component items that belong to the end items undergoing repairs at the depot.

Because the transaction history is not clear and common to all in this process, the depot may not always know if a component item belonging to an end item was evacuated for repair within the operating forces prior to the end item being shipped to MDMC for depot-level repair. The depot will at times have to purchase these component items from the manufacturer in order to have a fully equipped item ready for shipment upon completion of depot-level maintenance. These component items are no little expense; some of them can exceed $100,000 in price (K. Luckie, personal communication, 29 January 2018). The strain this can put on the system can be extensive. A study looking at the degree to which this and issues like it are occurring and having an impact on the costs of depot-level maintenance in the Marine Corps would be greatly beneficial to the understanding of all the challenges that must be addressed by the depot-level maintenance of not only the LAV, but all principle end items throughout the Marine Corps. While these are only a few examples of follow-on work, there are numerous potential applications for the DES model at MDMC that could ultimately provide the Marine Corps with additional ideas and methods for improvement of its maintenance processes.

THIS PAGE INTENTIONALLY LEFT BLANK

# LIST OF REFERENCES

Almeder, C., Preusser, M., & Hartl, R. (2009). Simulation and optimization of supply chains: alternative or complementary approaches? *OR Spectrum, 31*(1), 95–119. https://doi.org/10.1007/s00291-007-0118-z

Bair, T. D., Belen, Jr., F. C., et al. (2017). *Marine Depot Maintenance Command strategic planning support report*. State College, PA: Penn State Applied Research Laboratory.

Buss, A. (1995). A tutorial on discrete-event modeling with simulation graphs. In *Winter Simulation Conference Proceedings, 1995.* (pp. 74–81). https://doi.org/10.1109/WSC.1995.478708

Buss, A. (2000). Component-based simulation modeling. In *2000 Winter Simulation Conference Proceedings (Cat. No.00CH37165) 1*(1), pp. 964–971. https://doi.org/10.1109/WSC.2000.899899

Buss, A. (2017). Discrete Event Simulation (DES) Modeling. [Unpublished course book]. Naval Postgraduate School.

Cigolini, R., Pero, M., Rossi, T., & Sianesi, A. (2014). Linking supply chain configuration to supply chain performance: A discrete event simulation model. *Simulation Modelling Practice and Theory*, *40*, 1–11. https://doi.org/10.1016/j.simpat.2013.08.002

Curling, T. (2016, September). *USMC inventory control using optimization modeling and discrete event simulation* (Master's thesis). Monterey, California: Naval Postgraduate School. Retrieved from https://calhoun.nps.edu/handle/10945/50528

Fujimoto, R. M. (1990). Parallel discrete event simulation. *Commun. ACM*, *33*(10), 30–53. https://doi.org/10.1145/84537.84545

JMP Pro Version 13.1.0. [Statistical analysis software package and documentation]. Cary, NC: SAS Institute Inc., 2016.

Kumar, S., & Nottestad, D. A. (2013). Supply chain analysis methodology—Leveraging optimization and simulation software. *OR Insight*, *26*(2), 87–119. https://doi.org/http://dx.doi.org/10.1057/ori.2012.10

Manuj, I., Mentzer, J. T., & Bowers, M. R. (2009). Improving the rigor of discrete-event simulation in logistics and supply chain research. *International Journal of Physical Distribution & Logistics Management*, *39*(3), 172–201. https://doi.org/10.1108/09600030910951692

Marine Corps Logistics Command. (2017). Logcom command brief [Unpublished presentation].

Marine Corps Order 4790.2C. (2012, December 17). Washington, DC: United States Marine Corps.

Mele, F. D., Guillén, G., Espuña, A., & Puigjaner, L. (2006). A simulation-based optimization framework for parameter optimization of supply-chain networks. *Industrial & Engineering Chemistry Research*, *45*(9), 3133–3148. https://doi.org/10.1021/ie051121g

Mullins, M., Adams, T., Simms., R. (2005*). Analysis of Light Armored Vehicle Depot Level Maintenance* (MBA Professional Report). Monterey, CA: Naval Postgraduate School. Retrieved from https://calhoun.nps.edu/bitstream/handle/10945/33970

National Research Council, Education, (1998). *Modeling human and organizational behavior: application to military simulations*. Washington, D.C.: National Academies Press.

Naval Postgraduate School SEED Center website. (n.d.). NOLHdesigns_v6 [Excel spreadsheet tool for NOLH generation]. Retrieved May 9, 2018, from https://my.nps.edu/web/seed

Power World Corporation. (2014). *Time step simulation* [Presentation]. Retrieved May 8, 2018 from https://www.powerworld.com/files/M07Time-Step-Simulation.pdf

Schlegel, M., Brosig, G., Eckert, A., Engelke, K., Jung, M., Polt, A., (…) Vogt, C. (2006). Integration of discrete-event simulation and optimization for the design of value networks. In W. Marquardt & C. Pantelides (Eds.), *Computer Aided Chemical Engineering* (Vol. 21, pp. 1955–1960). Elsevier. https://doi.org/10.1016/S1570-7946(06)80334-2

Schruben, L. (1983). Simulation modeling with event graphs. *Communications of the ACM, 26*(11). pp. 957–963.

United States Marine Corps. (1992, November 2). Principles of Inspect, Repair Only as Necessary (IROAN) procedures and preparation of IROAN Publications. Washington, DC: Department of Defense.

# INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California