NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

EFFICIENCY TESTING IN A HUMAN-CENTRIC WORKFLOW: THE UNITED STATES ARMY TACTICAL CYBER REQUEST PROCESS

by

Anna C. Hughes

June 2023

Thesis Advisor: Susan M. Sanchez
Second Reader: Ross J. Schuchard

THIS PAGE INTENTIONALLY LEFT BLANK

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

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2023	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE
EFFICIENCY TESTING IN A HUMAN-CENTRIC WORKFLOW:
THE UNITED STATES ARMY TACTICAL CYBER REQUEST PROCESS

5. FUNDING NUMBERS

6. AUTHOR(S) Anna C. Hughes

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

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

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

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

The importance of United States (U.S.) Army Cyber operations is constantly increasing, along with the breadth of large-scale combat operations environments that must be considered. Traditionally, cyber operations support has been seen as a strategic asset. The U.S. Army Cyber Command (ARCYBER) recognizes the need for tactical commanders to employ cyber effects and published an updated policy to allow tactical commanders to submit cyber support requests. Handling these requests effectively is important due to the far-reaching effects of cyber targeting and the limited capacity of cyber support resources. Efficiency, represented as the timeliness of processing requests, is the primary measure of effectiveness. Computational modeling provides an avenue to generate and process over one million requests within minutes while comparing different variants of the process, rather than waiting for lessons to be learned in the field. We created a simulation model to represent this request process, while including random variation in the proficiency and learning behavior of support teams, and then conducted structured testing through designed experiments to yield insights about the process performance. Request service times, arrival rates, starting proficiency, and learning curves play significant roles in overall efficiency. We recommend further experimentation once more data is collected. Our approach serves as a foundation for modeling human behavior effects in similar studies.

14. SUBJECT TERMS cyber, process management, simulation, human behavior, model, data farming, Army Cyber Command, ARCYBER	**15. NUMBER OF PAGES** 71
	16. PRICE CODE

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

NSN 7540-01-280-5500

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

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

EFFICIENCY TESTING IN A HUMAN-CENTRIC WORKFLOW: THE UNITED STATES ARMY TACTICAL CYBER REQUEST PROCESS

Anna C. Hughes
Captain, United States Army
BAS, University of Arizona, 2014
MBT, University of Georgia, 2020

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
June 2023

Approved by: Susan M. Sanchez
Advisor

Ross J. Schuchard
Second Reader

W. Matthew Carlyle
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The importance of United States (U.S.) Army Cyber operations is constantly increasing, along with the breadth of large-scale combat operations environments that must be considered. Traditionally, cyber operations support has been seen as a strategic asset. The U.S. Army Cyber Command (ARCYBER) recognizes the need for tactical commanders to employ cyber effects and published an updated policy to allow tactical commanders to submit cyber support requests. Handling these requests effectively is important due to the far-reaching effects of cyber targeting and the limited capacity of cyber support resources. Efficiency, represented as the timeliness of processing requests, is the primary measure of effectiveness. Computational modeling provides an avenue to generate and process over one million requests within minutes while comparing different variants of the process, rather than waiting for lessons to be learned in the field. We created a simulation model to represent this request process, while including random variation in the proficiency and learning behavior of support teams, and then conducted structured testing through designed experiments to yield insights about the process performance. Request service times, arrival rates, starting proficiency, and learning curves play significant roles in overall efficiency. We recommend further experimentation once more data is collected. Our approach serves as a foundation for modeling human behavior effects in similar studies.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ARCYBER	U.S. Army Cyber Command
BDE	brigade
CEMA	cyber electromagnetic activities
CERF	cyber effects request form
CO-IPE	cyberspace operations-integrated planning element
COCOM	combatant command
DA	Department of the Army
DIV	division
G3	operations staff; division and higher echelons
JCC	joint cyberspace center
JCS	Joint Chiefs of Staff
JFHQ-C	joint forces headquarters-cyberspace
JTF	joint task force
LSCO	large-scale combat operations
MOE	measure of effectiveness
OCO	Offensive Cyber Operations
OE	operational environment
PCS	permanent change of station
RFS	request for support
RMSE	root mean square error
S3	operations staff; battalion and brigade
S6	signal and communications staff; battalion and brigade
USCYBERCOM	United States Cyber Command
USSTRATCOM	United States Strategic Command

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

The need for rapid, flexible cyberspace operations support is growing in the United States (U.S.) Army. This creates a potential for a rapid increase in demand in cyberspace operations support from the user to the enterprise level. The U.S. Army Cyber Command (ARCYBER) recognizes this need and has adapted policy to allow tactical commanders, to the brigade (BDE) level, to submit offensive and defensive cyber support requests. To keep pace with the prevalence of the cyber domain, an implied requirement exists to employ efficiency testing on these processes to ensure that ARCYBER can adjust procedures faster than the pace of government policy revision.

These are relatively new processes with little data collected beyond anecdotal measures. This study uses a carefully constructed simulation model to recreate a cyber support request process that mimics the formally published workflow. A data farming approach produces an extremely large, designed dataset of simulation output for exploratory analysis. This model applies the reported anecdotal data of the length for the entire request process combined with several realistic input parameters based on knowledge of general business practices of service members within different organizations in the U.S. Army. For this study, efficiency is considered as pressuring the average time that it takes for a request to pass through the approval process, or a request's time-in-system, to be as low as possible.

Exploratory analysis of the farmed dataset reveals several key findings about this specific offensive cyber request process. The efficiency of this process is largely influenced by amount of time that each team spends on processing an individual request and the rate at which these requests enter this process. One person per organization with this task generally proves ineffective. Assigning two people per organization lowers the average time to process requests, with specific conditions of having these individuals available at least 57% of the time and having some level of proficiency, or ability to learn quickly, about processing requests. These service members increase their time to process requests when these conditions are not met, or when support requests arrive quickly as two per day. A third person added to this team has little impact on the average time to process requests,

but it does make the processing time more consistent. It is likely that removing one approval authority in this process will reduce the overall processing time for requests, but further experimentation is required to determine how significant the impact is.

The service time distribution used in business process modeling has significant impact on measuring a request's time-in-service. Since there is no data provided on how long each organization spends on producing, revising, or approving the cyber support requests, there is not a known mean service time range nor distribution for modeling the average time that it takes for a given individual or team to complete this task. Thus, four service time distributions are modeled and compared in this request process. The main recommendation to address this challenge is to focus on data collection of the service times for each organization in the request process. We encourage future simulation experiments to explore multiple service time distributions, including those not found in this model, to assess the performance and robustness of this request process and similar processes.

This study demonstrates the ability to provide insights on efficiency improvement within the cyber request support process through the use of simulation, data farming, and incorporating randomness with quantitative effects from human behavior, even without much data collected from the existing process. Streamlining request approval ultimately provides tactical commanders with support more quickly, enabling them to turn their focus away from process management and towards operations planning and execution. Methods similar to those used in this study can be applied in any effort to give time back to commanders, leaders, and warfighters for a more efficient, fully supported fighting force.

ACKNOWLEDGMENTS

My most gracious thanks belong to my family. To my husband, Johnnie, and my daughters, Haley and Jordyn: thank you for moving away from your friends and family for two years on this California journey. Thank you for your time spent sharing our hardships and celebrating our accomplishments over these last two years. I appreciate every hug, every smile, every tear, and every second that we got to enjoy this experience together. To Haley: I know graduating was rushed and stressful for you. I'm so proud that you! Letting you leave the nest early is bittersweet, but you're ready to start flying as long as you come home for breaks and call me every day! To our children, Sydney, Kai, and AnnMarie: we miss you and have never stopped cheering for you, even 2,800 miles away. We can't wait to see you again!

To Professors Susan and Paul Sanchez: you have dedicated countless hours over the past eight months to my success in this endeavor. Thank you so much for your time, mentorship, and laughs. To Dr. Susan Sanchez: I hope your next journey is as fulfilling and satisfying as this one was. May you and Paul enjoy the fruits of your labor!

To LTC Ross Schuchard: thank you infinitely for your support and mentorship through the last two years. Thank you for volunteering benevolently as my thesis reader and for your mentorship over a host of topics as wide as the electromagnetic spectrum.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The role of cyberspace operations is rapidly increasing in our modern world. As the use of the cyberspace domain increases in the public and private sector with the demand for cyber security in these sectors, increases with it. Adversaries find new methods of targeting while we fight on multiple fronts to protect our information and our networks.

In the military, across all branches, we typically view cyber operations as strategic level defense high above the individual warfighter (White 2019). This is due to the incredibly far-reaching effects that can arise from compromised data. The United States (U.S.) Army is on the forefront of introducing cyber operations capabilities to tactical commanders to leverage in offensive and defensive measures. ARCYBER recently published a formalized process for tactical commanders to request operational support for different cyber effects (Department of the Army [DA] 2021). The intent is to make cyber operations more effective and more accessible, as well as raise awareness of using cyberspace operations as a force multiplier.

A. PROBLEM OVERVIEW

Implanting a strategic-based process in a tactical environment comes with a variety of issues as well. Tactical commanders usually make decisions in austere environments with as much information as they have on-hand; speed and efficiency are essential for a higher probability of mission success (Senft 2022). Strategic level processes are centered around compliance, thoroughness, and defined approval channels (White 2019). Mixing these different approaches can yield severe frustration with the system for both tactical leaders tasked with incorporating cyberspace operations and for the approval chain tasked with allocating resources to support these asset requests.

Any new process comes with a period of trial and error in implementation. While an organized effort to provide tactical commanders with new assets is vital to maintaining momentum, using trial and error methods will yield antiquated results, even if the process is perfected tomorrow. The cyberspace domain is constantly changing. It is difficult to rely on policy or Department of Defense (DOD) internal processes to stay ahead, or even

abreast, of rapidly evolving technologies without a faster method to test vital processes for efficiency; "in short, the DOD is too slow for cyber" (Schoeni 2017, p. 834).

B. PROPOSED SOLUTION

This thesis explores efficiency testing in one category of cyber support requests, Offensive Cyberspace Operations (OCO), from a tactical BDE staff through the formal approval chain prescribed in Field Manual 3-12, *Cyberspace Operations and Electromagnetic Warfare*. This specific process is less complex than the Defensive Cyberspace Operations (DCO) support requests. The process is simulated while varying parameters that revolve around how service members may interact in a business process environment. Since there are no existing statistics recorded for how the OCO support requests traverse this process, we farm data using the results of a designed experiment run through a discrete event simulation model. The results are analyzed for insights and observations about the OCO request process.

C. ASSUMPTIONS AND LIMITATIONS

This thesis is designed to study the different interactions between and within organizations in this process, and how these interactions contribute to the overall processing time of an OCO support request. Any insights can be used to direct more attention to certain areas of the approval chain, to identify factors that affect the overall processing time, or for making general observations about business processes in similar environments. This research is not intended to optimize the process or make assessments about specific units in the approval process.

For this model, all OCO support requests are ultimately approved by each approval authority. No request is denied, so that all requests are approved with an opportunity to cancel after the request later in the process by the requesting unit. Each request is the same priority; thus, no request is "fast-tracked" to purposefully force it through the process faster than any another request. There is also no warm-up period for this model. Instead, there is a ramp-up period as requests start arriving, followed by a ramp-down at the end of the year (as the specified time period). The requesters are not interested in submitting new requests

at the end of the year, but they are waiting to receive approval from requests that were submitted earlier in the year.

D. RESULTS OVERVIEW

The teams assigned to process OCO support requests in each unit ideally is comprised of two people who can maintain their collective availability at 57%. This team can support processing requests consistently while requests arrive to the team at a rate up to one request every day. We can estimate the amount of time that a team will take to process requests when we wholistically consider the team staffing, their availability, their proficiency and learning capacity, and the arrival rate of these requests to the team. If there is an opportunity to cut down on at least one approval authority in this process, then the average time to process requests is highly likely to decrease. We recommend a focused effort on data collection in this process for future experimentation and more refined insights, specifically on the service times to submit, revise, and approve these requests to determine the distribution of these service times.

Introducing variability into this process model, and other process models involving human interaction, is important to providing a more realistic model and more thorough analysis. Quantifying these behaviors as inputs, structural components, and variables has significant impact on this research. The time to process requests is significantly affected by the factors that vary due to the randomness in the proficiency and learning curve within the model. We recommend that methods similar to those in this study be used for future research in other processes and workflow testing.

E. THESIS ORGANIZATION

This thesis is organized into four subsequent chapters. Chapter II discusses U.S. Army Cyber operations development and the importance of the human factors involved in this process. Chapter III outlines the simulation model structure and development. The results in Chapter IV include analysis of the farmed data with observations and additional experimentation. Chapter V summarizes our research findings, recommendations, and proposed future work.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND AND LITERATURE REVIEW

This chapter provides an overview of U.S. Army Cyber operations development and tactical operations cyber support. We highlight the importance of acknowledging and including factors that represent human behavior. We also briefly introduce and discuss the modeling approach used for this study, which is described in more detail in Chapter III.

A. U.S. ARMY DEVELOPMENT OF CYBERSPACE OPERATIONS

Army cyberspace operations play an essential role in unified land operations, with the employment of cyberspace capabilities in missions where cyber superiority is the main objective as well as integrating cyber capabilities into overall mission planning (Joint Chiefs of Staff [JCS] 2018). ARCYBER was originally formed as a strategic capability with the establishment of United States Cyber Command (USCYBERCOM) under the United States Strategic Command (USSTRATCOM) in 2009 (Hollis 2010). As the Army shifts from conducting counterinsurgency operations to large-scale combat operations (LSCO), it is also redefining how it employs cyber capabilities. There is a change in the Army's approach to cyber from being a strategic, technical asset to an integrated planning capability alongside the more traditional "intelligence, signal, information advantage activities, space, and fires capabilities" (DA 2021) at the tactical level. In this sense, the Army is actively adapting processes to support tactical units with the regular review and update of the cyberspace operations doctrine.

At its core, the U.S. Army's mission is to fight and win wars through ground combat engagements (White 2019). The Army has an especially complex mission because the land domain addresses the presence of humanity, including culture, religion, and politics, which affects nearly all aspects of ground combat. "The primacy of the individual soldier, and the reality of ground combat as an intimate human endeavor, has left the Army with a healthy institutional skepticism towards the promises of new technology" (White 2019). This human-centric focus creates an inadvertent difficulty to embrace abstract, technological environments, such as cyberspace. Thus, changing the Army's approach to cyberspace

heralds an organizational culture shift in the way that units and service members plan for, use, and integrate cyber capabilities into current operations.

B. ACCEPTING AND INTEGRATING TACTICAL CYBER OPERATIONS SUPPORT

Units at the tactical level, specifically corps, division (DIV), and BDE, face this complexity in operations every day. Tactical unit commanders have reported their attempts to employ cyberspace operations at the tactical level as difficult, at best (Senft 2022). This naturally feeds into their developed distrust of integrating new capabilities, which can result in leaving cyberspace out of planning considerations altogether (White 2019). To ease the healthy skepticism of new technology and encourage commanders of tactical units to integrate cyberspace operations, an officially defined process for these tactical units to request cyberspace operations support was published in August 2021 (DA 2021). It describes two main types of cyber support requests, which are OCO and DCO, with the routing processes for requests. The specific form, the cyber effects request format (CERF), is composed by the requesting unit and routed to the Joint Force Land Component Command (JFLCC); it is then transposed into a request for support (RFS) to route at higher echelons.

Several leaders within ARCYBER report that a specific OCO process, targeting, takes six months once the CERF is successfully passed out of the tactical unit to the joint level as an RFS. Many tactical unit commanders view six months as lengthy and the process as too cumbersome (Senft 2022). Figure 1 displays the OCO support request process, which is the focal point for this study.

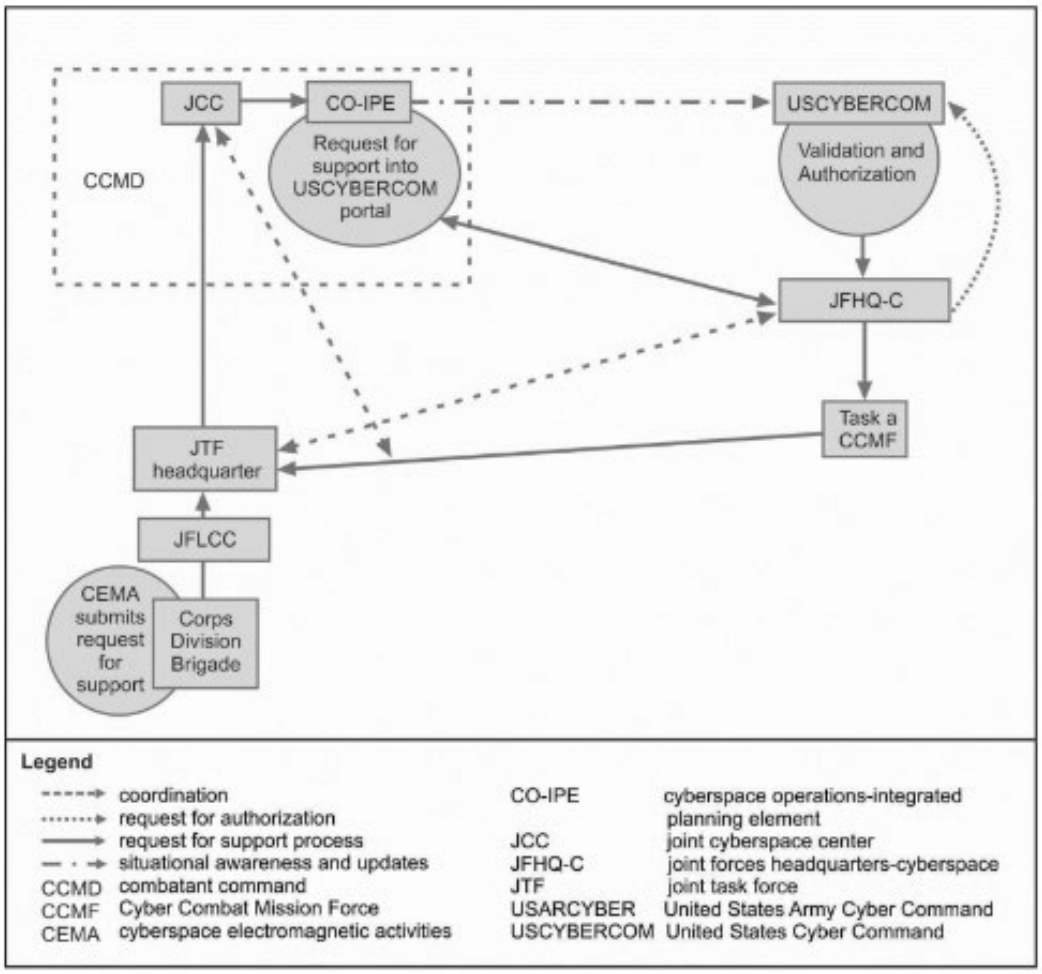


Figure 1. Routing Process to Request Offensive Cyberspace Operations Support. Source: DA (2021).

C. PLANNING FOR HUMAN INVOLVEMENT IN PROCESSES

Understanding humanity in warfare, planning, and conducting routine processes is vital to how receptive the unit and service members are to integrating cyber capabilities into tactical operations (Connable et al. 2018). Capturing the effects of humanity is difficult without honest self-reporting, which is difficult to achieve due to the participants' desire to overstate what they believe are correct, preferred, or popular behaviors (Baird and Özler 2012). Unit surveys and behavioral observations present issues such as the Hawthorne Effect; this accounts for changes in behavior when subjects are aware that they are part of an experiment (Merrett 2006). Therefore, this study uses two measures as the primary

effects of human interaction on the request process to quantify and dramatically simplify them. These are two principal behaviors that the U.S. Army uses as performance metrics: availability and proficiency.

1. Availability

Personnel availability is a vital metric that military units use to measure overall operational readiness. This study uses the availability for each unit in the OCO support request process chain-of-custody to determine how long that unit will take to process their portion of the request. Low availability extends the time that it takes to process a request. Most service members have primary tasks as well as additional duties; a primary task is directly related to the service member's position within an organization, and an additional duty is a specific, directed responsibility assigned to the service member beyond the scope of their primary tasks. Processing an OCO support request is not a primary task for any of the organizations in the chain-of-custody. It is assumed that primary tasks are completed with a higher priority, thus extending the delay for processing any OCO support request.

2. Proficiency

The U.S. Army measures proficiency in a task based on specific descriptions of the operational environment (OE) condition when and where the task is conducted. The task for an individual or unit is assessed with a rating of "Trained, needs Practice, or Untrained" (T-P-U) based on their completion of said task under the specific OE conditions in Figure 2.

Plan and Prepare					Execute						Assess
Operational Environment			JLTX	Training Environment (L/V/C/G)	% Leaders Present at Training/ Authorized	% Present at Training / Authorized	External Eval	Performance Measures	Critical Performance Measures	Leader Performance Measures	Task Assessment
Dynamic & Complex	Night	Hybrid Threat	Yes	Proponent Establishes Training Environment Standards (FTX, STX, CPX, STAFFEX, TEWT, etc)	≥85%	≥80%	Yes	≥90% GO	All	≥90%	T
Dynamic or Complex					75-84%			80-89% GO		80-89%	T-
Static and Simple	Day	Regular or Irregular Threat	No		65-74%	75-79%	No	65-79% GO			P
					60-64%	60-74%		51-64% GO	<All	<80%	P-
					<60%	<60%		< 50% GO			U

Figure 2. Objective Task Evaluation Criteria. Source: Hoffman (2015).

This study uses a simplified definition of proficiency from Table 1 when evaluating a unit's proficiency rating in the OCO support request process. We consider each unit with a collective proficiency of individuals having processed OCO support requests. For example, if the requesting unit has mostly team members that processed several to no OCO support requests, they may have a unit proficiency of "needs Practice;" "needs Practice" is considered proficiency values from 60% to less than 75%.

Table 1. Proficiency Simplified. Adapted from Hoffman (2015).

Qualitative Proficiency Value	Quantitative Adaption in Model
Untrained	[0, 0.6)
Needs Practice	[0.6, 0.75)
Trained	[0.75, 1)

D. MODELING THEORY

Data farming is commonly used in conjunction with simulation modeling to answer questions that have not been addressed previously. It allows events with uncertain,

numerous possible outcomes to be examined so that "overall and unexpected results may be captured and examined for insights" (Horne and Seichter, 2010). NATO's Modeling and Simulation Group (NMSG) conducted a three-year task group to assess data farming in proof-of-concept case studies for military decision support (Horne and Seichter, 2010). The NMSG later applied it with an agent-based simulation model for analysis military decision support for the German Federal Armed Forces to "find a robust configuration of a combat outpost (COP) against different kind of threat scenarios" (Kallfass and Schlaak, 2012). This study applies similar methods to the developed discrete event simulation model herein. The fundamentals associated with each of these subjects are addressed in this section.

1. Tandem Queuing System with Discrete Event Simulation

A queuing system represents a system with a service facility at which some type of units, usually "customers," arrive for a service to be performed by one or more "servers." This study uses "requests" as the arriving units and "units" as the servers. A tandem queue occurs when the departing "requests" from one system becomes the arrival "requests" for the next system. Discrete event simulation is used to schedule events, ensuring that the "requests" through the queuing system and that the individual events occur according to their specified order. Figure 3 depicts an event graph, or a visual representation of a queuing system, of a tandem queue with one overall queue system divided into two sub-processes, *Q1* and *Q2*. A "request" arrives to *arrival 1* at its respective arrival time, then moves through the sub-process *Q1*. The same "request" then becomes an arrival to sub-process *Q2*.

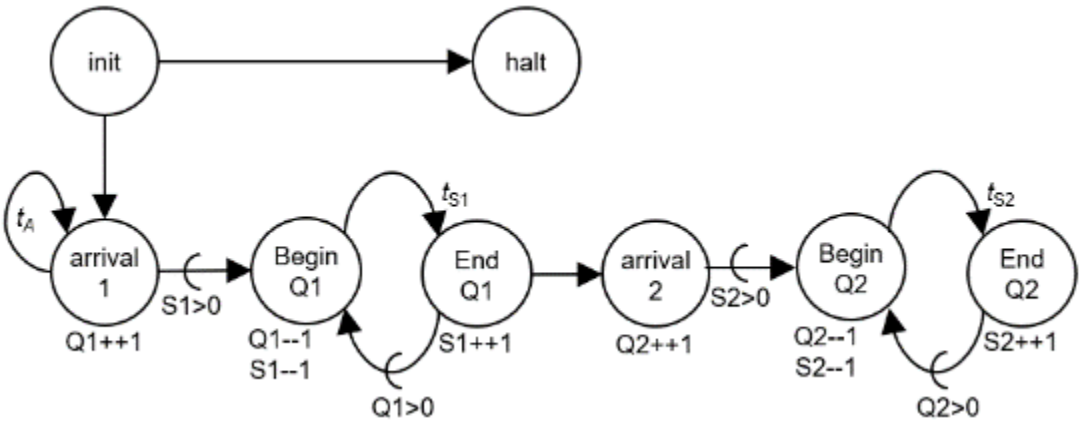


Figure 3. Adaptation of Model Event Graph to Display Tandem Queue Only

2. Data Farming

The officially released process for tactical OCO support requests is relatively new (DA 2021), with no known data collection so far. To learn about this simulation model's behavior, we must construct, or "farm," data using design of experiments. Using "farmed" data often helps when identifying useful information, such as "the cause-and-effect relationships between the model's input factors and the simulation output" (Sanchez 2018). The design of experiments uses stacked nearly orthogonal Latin hypercubes in this study to structure the experiment with carefully chosen permutations of factor settings while providing flexibility for constructing metamodels (that is, statistical models of the input/ output behavior) that reflect the complexity of the simulated system.

THIS PAGE INTENTIONALLY LEFT BLANK

III. SIMULATION MODELING

This chapter provides the details of the study's model formulation in Python, including inputs, variables, and output types. We discuss the transformation from workflow graph to event graph and the methodology behind the model's design. Lastly, we describe the farmed dataset with the process to attain similar datasets for future research.

A. MODEL PARAMETERS AND VARIABLES

The simulation model is written in Python, uses the SimpleKit.py (Sanchez P 2007) Python module and adapts tandem.py (Sanchez P and Sanchez S 2023) for functionality with YAML for managing parameters across multiple experiments. YAML is a data-serialization language that is easily read by humans; it is formerly known as *Yet Another Markup Language* and later transformed to *YAML Ain't Markup Language*.

The model depicts the path of an OCO support request from a BDE Cyber Electromagnetic Activities (CEMA) and Signal and Communications (S6) staff, through their higher headquarters' Operations (S3/G3) staff and strategic approval levels, then back to the requester. Table 2 describes each of the input parameters. Table 3 describes each of the state variables in the model; state variables are used internally to the model for tracking performance statistics.

Table 2. Model Parameters

Parameter Name	Description with Baseline Input
`queue.maxServers`	The maximum number of people in a unit assigned to process OCO support requests at any given time.
`queue. svcTimeMean`	The average amount of time, in days, that a given unit spends processing an OCO support request if all people in the unit are focused solely on OCO support request tasks.
`queue.svcDist`	The shape of the service time distribution. Possibilities include a shifted exponential, triangular, uniform, or exponential distribution.

Parameter Name	Description with Baseline Input
`queue.next`	The next unit to process the request.
`queue.avail`	A number between 0 and 1 which modifies the (randomly generated) time required to process a request by adding (1 - availability) to the queue scheduling delay; if `queue.avail = 1` there is no change. For example, if `queue.avail = 0.5` the processing time will increase by 0.5 days.
`queue.learnCurve`	A set of 100 x- and- y values of a randomly generated unit learning curve. These curves are sigmoid cumulative distribution functions (CDFs) which represent the performance from success-based learning (Leibowitz et al. 2010).
`queue.proficiency`	A number between 0 and 1 which represents the starting proficiency of a given unit; it is a random y-value on the unit's learning curve. Units increase proficiency when a request is returned for revision and decrease proficiency when a service member leaves the unit.
`queue.name`	The name of the current queue.
`queue.retainRate`	A number between 0 and 1 which represents the retention rate of the unit processing the request. If a member of the unit leaves, then the unit proficiency decreases slightly. *`proficiency*(1-maxServers)/maxServers`*
`queue.incrementMin`	An integer between 0 and 99 which represents the lower value of the range to increment the x-value on the specific queue's learning curve when a revision occurs.
`queue.incrementRange`	An integer between 1 and 99 which represents the upper value of the range to increment the x-value on the specific queue's learning curve when a revision occurs.
`request.arrRate`	The arrival rate of the request; this represents how often, on average, the BDE S6/CEMA staff submits an OCO support request.
`shutdownTime`	The time at which no more requests are made, and all other requests in the model are processed until cancelled or completed.

Most of the model parameters in Table 2 are self-explanatory. However, a few merit further explanation. First, the `queue.svcTimeMean` parameter represents the average hands-on time that is required for processing OCO support requests at a specific unit. This is not the same as the time a request spends in that particular queue because it may have to wait for other requests to be completed before it reaches a server.

Second, the potential service time distributions are automatically parameterized by a single input parameter, the service time mean/rate for an exponential distribution, as in Sanchez and Sanchez, 2020. All of distributions used have the same mean, and the variances are the same with the exception of the exponential distribution. These distributions shapes are shown in Figure 4.

Third, there are different ways that server availability can be implemented. One which we did not use is to randomly generate times when an available server becomes unavailable, or vice versa, similar to the way that random breakdowns are scheduled in models of manufacturing processes. We chose a different modeling method that shifts the entire service time distribution based on the availability. This method mimics the team members shifting their available time between completing primary tasks and additional duties discussed in Chapter 2 while balancing a heavier workload due to team members being unavailable. When `availability=0`, it does not mean that the request will never be processed. It means that there is a significant delay in processing the request with an increase in the mean service time, which makes it much less likely that any specific request will be processed quickly.

Figure 4 demonstrates how this definition of availability affects the service times over each distribution. Here, the mean service time is 0.333; this is equivalent to eight hours of hands-on service time of a unit with the OCO support request. Each service time range over the respective distribution is identical in shape: only the scaling differs. Theoretically, the right tails of the exponential and shifted exponential distributions are unbounded, but for practical purposes we plot over more limited ranges. When availability=1, on the left in Figure 4, the service time range in the plot is [0,2]. On the right of Figure 4, there is a significant increase in the service time range, from [0.5,2.5], when the availability is reduced to 0.5. In this sense, the service time to process the request

remains the same while the availability of team members determines how fast one of the team members starts the task.

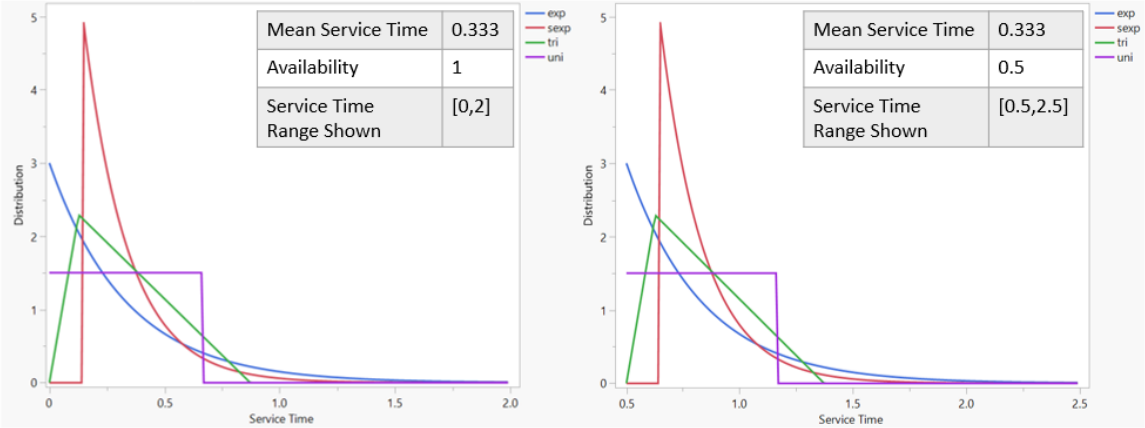


Figure 4. Service Time Distributions for Different Availability Levels

Finally, we provide more detail about the implementation of learning curves. Each queue models a group of service members within an organization who are assigned to revising, processing, and approving OCO support requests. The model randomly generates sigmoid cumulative distribution functions (CDFs) with 100 random numbers from one logistic distribution for each tandem queue; there is a new learning curve for each queue when each design point is created for each run. These sigmoid CDFs represent the group's learning curve regarding how to process the OCO support request. Each queue is then assigned a beginning proficiency as an (x,y) coordinate that exists on their respective sigmoid CDF. Throughout each replication, the group's proficiency grows each time a request revision occurs by moving up a random number of x-indices, ranging [`queue.incrementMin`, `queue.incrementRange`], forward on the x-plane while their new proficiency becomes the corresponding y-value.

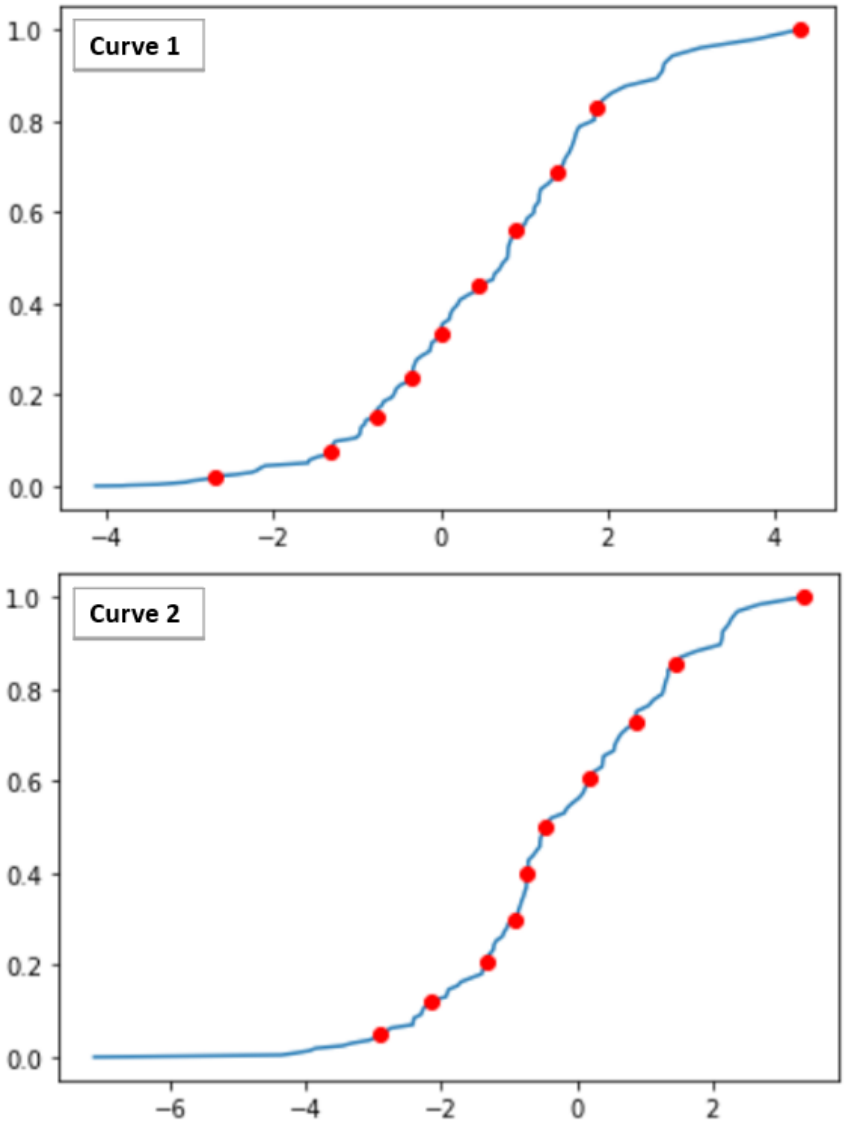
The group's proficiency decreases when a trained service member leaves. The decreased amount is dependent on the `maxServers` and represents the proportion of personnel that remain in the group after one trained service member leaves. For example, if `maxServers=3` and one service member leaves, the proficiency is multiplied by 2/3.

The x-value does not decrease in order for the proficiency to rise quickly after one revision, just as it would in a group that is already mostly proficient and trains a new service member how to complete a support request.

There are also cases when `maxServers=1` and the position is gapped, or no one is left to complete the task when a trained service member leaves. This decreases the proficiency to zero until one revision is completed so the learning curve may continue the established path for that specific replication. In reality, all the previous knowledge does not disappear if a position is gapped. Instead, the team finds an example request either in the office, a continuity document, or in reference materials and builds on that information. However, there is at least one revision to establish the team's initial knowledge base.

Figure 5 displays two example sigmoid learning curves which are generated in the same method as in the simulation. Each red point indicates an increase in ten x-indices across both curves. As x is incrementally increase on each curve, we see that most of the points lie in the middle of the "S," where the end-points of x [0] and x [99] are outliers which create the top and bottom flattening effect of the "S." We increase proficiency with a variable range of x-indices in the model to introduce an additional layer of variability, since some teams may travel faster along their learning curve than other teams. Figure 5 shows incremental increases to clarify how the movement along this curve may vary from one queue to another.

Curve 2 in Figure 5 has a steeper learning curve which rises before Curve 1. The steepness of the learning curve can be measured by the increasing y-values; y-values at the selected x-indices in Curve 2 are higher than those of Curve 1. The earlier increase in Curve 2 is apparent because the range of x-values is overall lower than that of Curve 1.



X Index	Y
10	0.021
20	0.077
30	0.152
40	0.239
50	0.335
60	0.441
70	0.560
80	0.690
90	0.831
100	1.00

X Index	Y
10	0.050
20	0.123
30	0.208
40	0.301
50	0.398
60	0.499
70	0.608
80	0.726
90	0.854
100	1.00

Figure 5. Examples of Randomly Generated Sigmoid CDFs within Model

Table 3. Model State Variables

State Variable Name	Description
`request.tMax`	The maximum time that a unit is willing to wait for a request to be approved. For each request, this is six months plus the time that the request takes to travel from the requester to the first strategic approval authority.
`request.arrivalTime`	The time that the request is initiated (arrives at first queue).
`request.timeInSystem`	The current time that the request has been in the system.
`request.ctRequest`	An integer incremented to track the number of requests in the model; used as a request identifier.
`request.ctRevise`	An integer incremented to track the number of times that a given request is revised.
`queue.learnRate`	An array that holds each learning curve value that the given queue uses throughout the lifetime of the run. The learning rate is determined using the start and end unit proficiencies.
`queue.learnIndex`	An integer between 0 and 99 to indicate which (x,y) pair the queue's learning curve uses. This coordinate is used for moving forwards or backwards when a unit revises a request or loses a team member, respectively.
`queue.ctCancel`	An integer incremented to track the number of times that a request is cancelled from a specific queue.
`queue.ctPCS`	An integer incremented to track the number of times that a queue has a team member that leaves the unit, thus decreases unit proficiency. A member commonly leaves due to a permanent change of station (PCS), but here PCS represents a service member leaving for any reason.

B. MODEL EVENT GRAPH

The simulation model is described by the event graph in Figure 6. Each circle represents an event that can be scheduled, with directed arcs that indicate a relationship between events. The request has a designated arrival time and moves through the events along the directed arcs. Each queue with "begin service" and "end service" events denote a unit in the OCO support request process. Requests may change route under conditions related to the submitting unit's proficiency or based on the time that the request has been in the system.

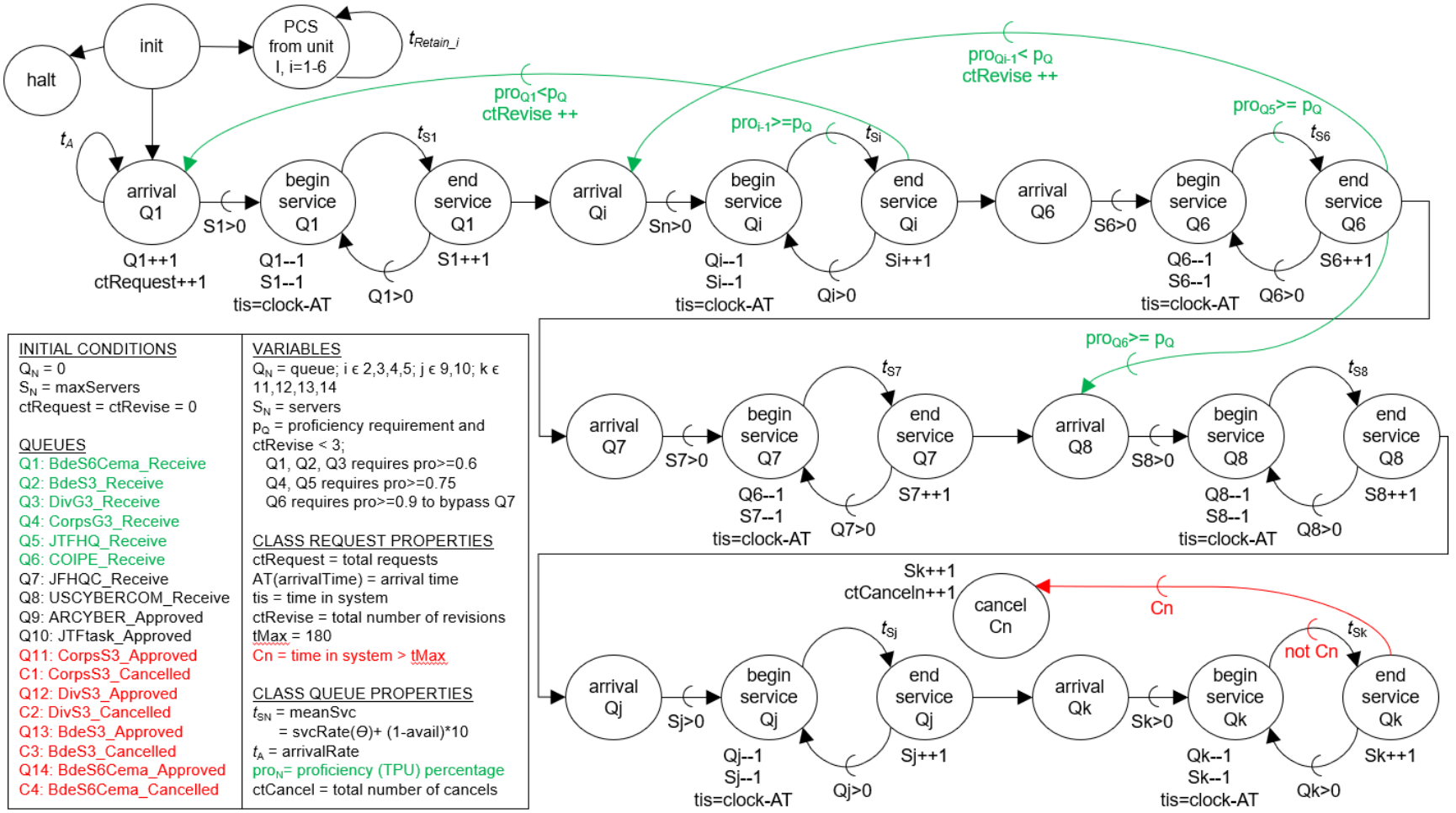


Figure 6. OCO Support Request Process Event Graph

1. Proficiency Related Conditions (Green)

Any relationship that represents a condition based on a unit's proficiency in Figure 6 is green; queues with this relationship are *Q1* through *Q5*. The first three queues (*Q1*, *Q2*, *Q3*) require a unit proficiency of at least 0.6 to successfully submit their request without revisions. These are all queues that represent units submitting the request to a tactical approval level (BDE to BDE, BDE to DIV, DIV to Corps). The queues *Q4* and *Q5* both require a unit proficiency of at least 0.75 because they are units submitting the request to a strategic approval level; it is assumed that there is a higher expectation of quality when submitting to the strategic units. Each time these unit proficiencies do not meet their respective minimum requirement, the request is returned for revisions after the higher-echelon unit reviews it with its respective service time. The revising unit increases its proficiency along its respective learning curve in a random increment, from one to 20, before it resubmits the request. There is a built-in exception here: if a request has been returned for revisions three times, then the request goes forward regardless of proficiency. This highlights a realistic frustration on the receiving end of the request to simply make the appropriate corrections and forward the request to the next unit.

One proficiency is unique in this set. The workflow depicted in Figure 1 has a specified indirect relationship between the cyberspace operations-integrated planning element (CO-IPE) and USCYBERCOM. In essence, if there is a strong professional relationship between these two organizations, then there exists an opportunity for the request to be passed regardless of any approval authority that lies in between. This behavior of bypassing one approval allows us to test whether W. Edward Deming's quality control observations in manufacturing, where reducing the inspectors of a product results in more accountability for each inspector and a higher-quality product (Suarez 1992), can apply to the increasing the timeliness of processing a request with multiple approval authorities. The unique relationship is presented in Figure 5 as well; if the *Q6* (CO-IPE) proficiency is greater than 0.9, then the request bypasses *Q7*, the joint force headquarters-cyberspace (JFHQ-C).

2. Time-in-Service Related Conditions (Red)

Cancelling a request for support is a common occurrence when the time waiting for approval has exceeded a subjective time set by the requesting unit. This maximum allowable time depends on how long the unit is willing to wait to receive approval before adjusting the plan to exclude or compensate for the support not being approved. It may even be the case that the planner or requester left the unit before the request was returned with approval and the position replacements do not want, need, or know about the request. Since an OCO support request is expected to take six months to approve, the maximum allowable time in this model, in days, equivalent to 180 days plus the time-in-service of the request at the last tactical echelon approval, the Corps G3 (*Q11*). Once a request is approved and the approval reaches *Q11*, then the `requestTIS` is tested against the maximum allowable time at *Q11* and each subsequent tactical unit to determine if the respective unit or original requester cancels the request.

$$C_n = \texttt{requestTIS} > \texttt{request.tMax}$$

C. CONFIGURATION FILES (YAML)

Several configurations are pushed to the model through a YAML file. The model is written to take YAML files as input from a command-line prompt to change variables' settings easily without making changes inside the Python file. Figure 7 displays a portion of the baseline YAML file used for this model. It creates a list of dictionaries with one value for each factor. These dictionaries are saved in the model, then used iteratively in series for the simulation model's input. We discuss this further in Chapter IV.

```
%YAML 1.1

# This configuration file tests OCOrequest.py with different request arrival rates and service time
distributions for each unit.

# The series uses 97 design points (3 stacks, 33 levels) and is generated from "stack_nolhs.rb". This
is an experiment design using pre-tabled nearly orthogonal latin hypercubes (NOHLs) with factor
scaling.

### arrivalRate: [0.143,2] to 3 decimal places

### service time distributions [1,4] integers

### svcDist:
###    1: sexp  # shifted exponential
###    2: tri   # triangle
###    3: unif  # uniform
###    4: exp   # exponential

### mean service time: [0.011,4] to 3 decimal place
### this is equivalent to a range of 15 minutes to four full days

### maxServers: [1,3] integers

### availability: [0.25,0.9] to 2 decimal places

### retainRate: [0.25,0.9] to 2 decimal places

### incrementMin: [0,20] integers

### incrementRange [1,50] integers
---

### DP 1
- arrivalRate: !!float 2.0
  svcDist: sexp
  meanSvc: 1.756
  maxServers: !!int 1
  availability: !!float 0.82
  retainRate: !!float 0.66
  incrementMin: !!int 14
  incrementRange: !!int 24
```

Figure 7. YAML File to Specify Inputs

D. MODEL OUTPUT STATISTICS

Table 4 describes the output statistics that represent the averages of variables in the model. These are captured each time an event is created along with several input conditions in one of many rows in a .csv file. When we conduct our experiments, we can compare the output statistics with their corresponding input parameters to identify what may be causing changes in the data. Other output includes the individual values for each queue during each request.

Table 4. Output Statistics

Statistic	Description
`requestTIS`	The processing time of the given request.
`avgProficiencyIn`	The average input proficiencies over each queue for each request.
`avgReviseOut`	The average revisions over each queue for each request.
`avgPCSOut`	The average team members that leave in each unit (queue) over the lifetime for each request.
`avgLearnRateOut`	The learning rate is determined using the start and end unit proficiencies. `learnRate [-1]/learnRate [0]`
`requestBypass`	The Boolean statement which represents if the CO-IPE unit proficiency was at least 0.9, inducing one unit to be skipped in the approval process.

E. EXPERIMENT DATA SET

The main design of experiment uses an NOLH with three stacks and 33 levels of eight factors, the request `arrRate` and the `svcDist`, `meanSvc`, `maxServers`, `availability`, `retainRate`, `incrementMin`, and `incrementRange` for each queue, to create 97 design points with 50 replications each. The entire experiment takes approximately 30 minutes to run using a Windows laptop with an i7-11800H processor, and it produces over 1.8 million request instances, each with their own unique output statistics. Table 5 describes the specific input factors and ranges or settings used to generate the set of farmed data.

Table 5. Factor Ranges and Settings for Designed Experiment

Factor Name	Description with Input
`request.arrRate`	[0.143, 2] to 3 decimal places This range is one request per week to two requests daily.
`queue.svcDist`	categorical input, coded as 1=shifted exponential, 2=triangular, 3=uniform, and 4=exponential for the distribution of `queue.svcTimeMean`
`queue.svcTimeMean`	[0.011, 4] to 3 decimal places. This range corresponds to mean service times from approximately 15 minutes to four full days.
`queue.maxServers`	[1, 3] to 0 decimal places (integer)
`queue.avail`	[0.25, 0.9] to 2 decimal places
`queue.retainRate`	[0.25, 0.9] to 2 decimal places
`queue.incrementMin`	[0,20] to 0 decimal places (integers)
`queue.incrementRange`	[1,50] to 0 decimal places (integers)

A scatterplot matrix of the design's quantitative factors appears in Figure 8. This design has seven stacks. The maximum absolute pairwise correlation among quantitative factors is 0.11, indicating that the design is nearly orthogonal despite the rounding required for specifying the `maxServers`.

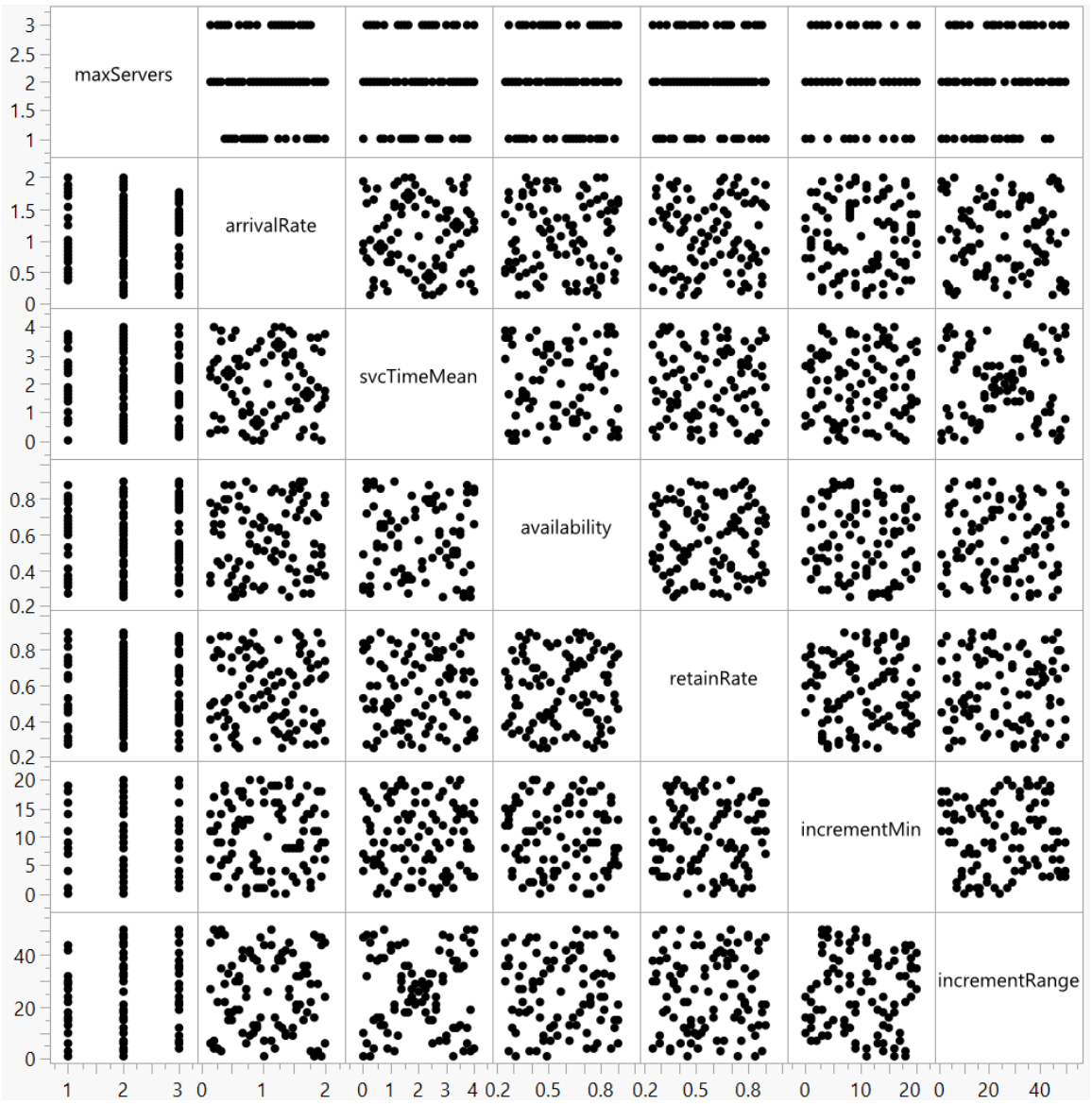


Figure 8. Scatterplot Matrix of Quantitative Factors

Varying the service time distributions provides us a unique insight; the exponential distribution is often modeled in an academic setting due to its desirable mathematical properties, but it is not often reflected in practice (Sanchez and Sanchez 2020). We can explore the significance of recording service times during the actual OCO request process in the future since we do not initially know what the service time distribution follows or if it is relevant to improving `requestTIS`.

There is only anecdotal data on how long any OCO support request takes to traverse this process, which is six months after it leaves the last tactical unit (the corps level) (Senft 2022). We conducted several preliminary experiments in the process of developing and verifying the simulation model. These experiments helped us identify ranges for the input parameters that seem realistic; it also provides a model where the raw data outputs the mean and the median `requestTIS` within 60 days of 190 days, which is a reasonable time close to the anecdotal known time of 180 days. This timeframe accounts for the estimated six months for approval with an additional 10 days to move from the original requester to the last echelon of the tactical unit, the Corps level.

IV. RESULTS

This chapter discusses various summary statistics for both ensuring the simulation model is a reasonable reflection of the OCO support request process and for gaining insights from the output data. We fit multiple regression metamodels to our farmed data to further discussion on important factors influencing our primary measure of effectiveness (MOE), `requestTIS`. The summary statistics use the raw datafile, with each request represented per row and a total of over 1.8 million requests processed. We also note the differences of mean and variance in design points by grouping the raw datafile by design point and replication; this is referred to as the grouped dataset.

A. REALISTIC MODELING WITH REASONABLE OUTPUT

Our first concern is reviewing summary statistics to create understanding about how the model compares to what we know about the actual OCO support request process. We want to be confident that the simulation has output statistics that mimic what we may see in this specific process, as well as any typical business process. This section reviews several of the simulation's output statistics to confirm that the model behaves as intended and is a reasonable reflection of the OCO support request process.

We begin by examining our output for our experiment's MOE, `requestTIS`. Since the distribution of `requestTIS` is skewed, we consider both the median and mean of the raw datafile. In Figure 9, the median `requestTIS` is 136 days and the mean is 250.15 days. This seems reasonable, while the extremely large range difference in days for extremely rare cases indicates either that there are other factors to explore regarding efficiency with `requestTIS`, or that there is a great deal of inherent variability in the model. We remind the reader that the output observations in Figure 9 are not independent and identically distributed but come from 97 different input factor configurations.

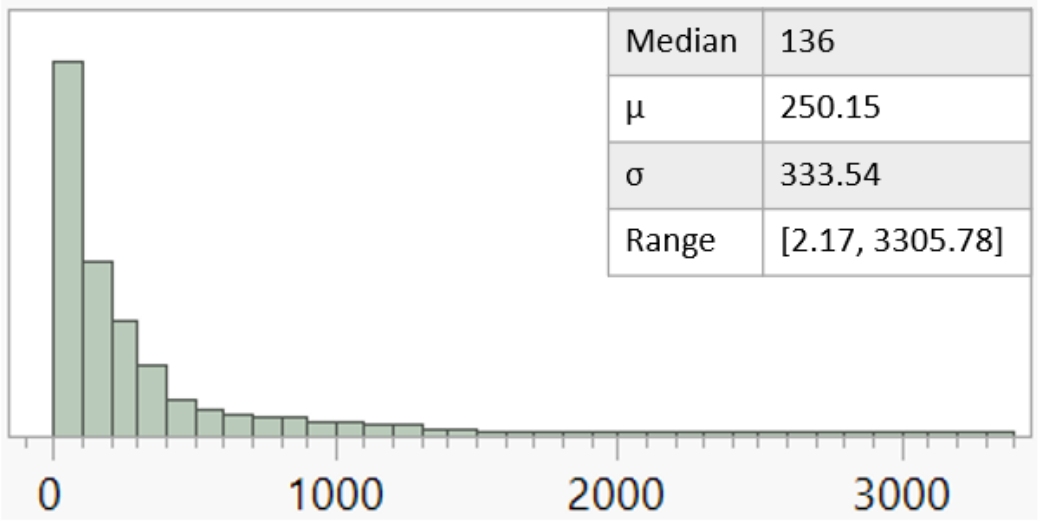


Figure 9. `requestTIS` Summary

Figure 10 depicts a histogram of `tMax`, which is also heavily skewed. The tactical unit only has three queues to move through before the additional 180-day process comprised of six queues. Interestingly, there are also no request cancellations in the output; every request comes back to the original requester in what the unit perceives as an appropriate amount of time. This suggests that most requests take a considerable amount of time to get through the tactical unit just to reach the strategic approval process. Tactical units typically spend a median time of 80 days and a mean time of 197 days processing the support request prior to submitting it to the first strategic echelon. Minimally, the tactical echelons unit can process the request in approximately half a day. The maximum observation of over 3,000 days implies that other factors may heavily influence `requestTIS`.

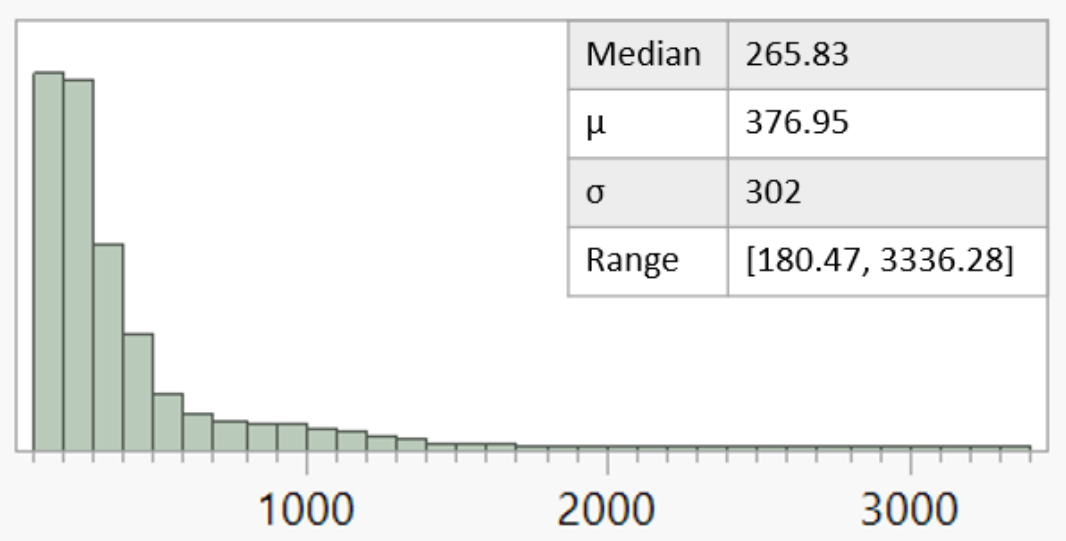


Figure 10. `tMax` Summary

The `avgPCSOUT` output, depicted in Figure 11, is influenced by `retainRate` and also influences the queue's randomly generated proficiency learning curve. Service members depart more often when `retainRate` is low, and the unit's processing proficiency decreases each time a service member departs. Approximately one service member per unit that is involved in processing requests leave during a typical request's lifetime. This is a reasonable number of average service members per unit over one year's timespan, and a very reasonable approximate range of [0,7] based on the wide range of `retainRate`.

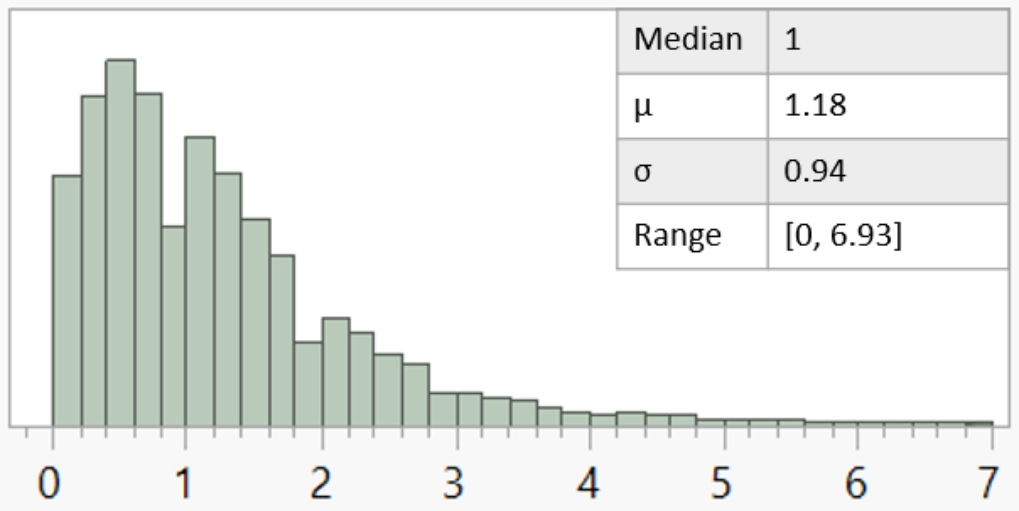
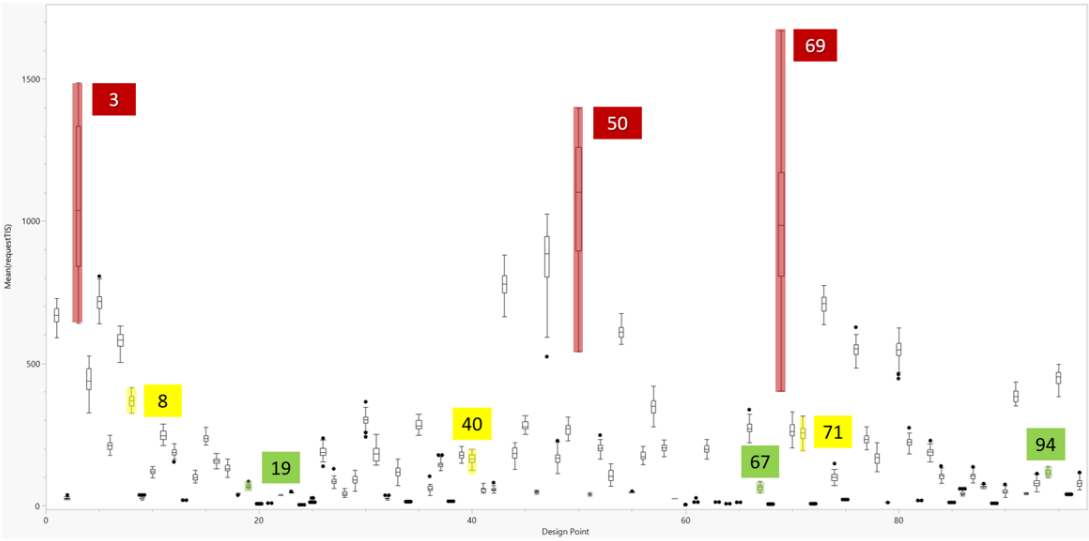


Figure 11. avgPCSOUT Summary

Figures 9–11 show that there is a great deal of variation in the handling of individual requests across the entire experiment. Different figures are needed to explore the differences by design point. Figure 12 illustrates the differences in the mean `requestTIS` by design point using the grouped dataset. The three design points with the highest corresponding mean `requestTIS` times are listed with three randomly selected design points with mid-range `requestTIS` times and three randomly selected design points with low range `requestTIS` times. All the design points with the highest `requestTIS` (red) have the most variability, relatively high `arrivalRate`, a low `incrementRange`, relatively high `svcTimeMean`, and `maxServers=1`. We observe that the design points with a mid-range `requestTIS` (yellow) have a declining `arrivalRate`, a declining `svcTimeMean`, a slightly increased `maxServers` range, and a significantly increased `incrementRange`. The design points with a low range of `requestTIS` (green) have very little variability, lower arrival rates, and a higher range for `maxServers`. From this preliminary view, we observe that the design points have a high variability, and that `requestTIS` trends lower with increasing `maxServers`, decreasing `arrivalRate`, and decreasing `svcTimeMean`.



requestTIS	DP	arrivalRate	svcDist	meanSvc	maxServers	availability	retainRate	incrementMin	incrementRange
High	3	1.768	tri	3.626	1	0.27	0.64	14	3
High	50	1.826	unif	3.252	1	0.49	0.45	11	1
High	69	2.0	tri	3.751	1	0.78	0.29	9	6
Mid	8	1.13	unif	3.751	2	0.84	0.41	4	50
Mid	40	1.013	exp	1.008	1	0.68	0.49	18	44
Mid	71	0.723	tri	1.756	1	0.35	0.82	1	27
Low	19	0.317	sexp	3.501	3	0.6	0.78	5	35
Low	67	0.375	tri	1.881	1	0.74	0.45	18	22
Low	94	1.652	tri	1.632	3	0.27	0.39	13	36

Figure 12. Mean `requestTIS` by Design Point, Comparison Chart

B. IMPORTANCE OF SERVICE TIME DISTRIBUTION

The service time distribution factor has a significant impact on `requestTIS`, especially when it interacts with other factors. One variable of interest over a request's lifetime, `requestBypass`, indicates whether one queue, the COIPE (*Q6*), has a high enough proficiency to bypass the JFTHQ-C (*Q7*). Figure 13 uses the grouped dataset to compare the mean `requestTIS` per design point by service time distribution in both the regular process (blue) and where one queue was bypassed (red). We observe that the median and range of the mean `requestTIS` for each service time distribution is lower when one approval authority is bypassed. In two cases, with triangular and uniform service time distributions, the medians of the means are very similar whether or not some requests are bypassed. The median of the mean `requestTIS` for the shifted exponential

distribution are only slightly different, and those of the exponential distribution vary the most. We can make an overall conclusion that bypassing one queue in this model significantly lowers `requestTIS`, and we can conclude that the action of bypassing one approval authority in this process improves overall efficiency. However, we cannot say how substantial the impact is without knowing which service time distribution resembles the service times in the actual process.

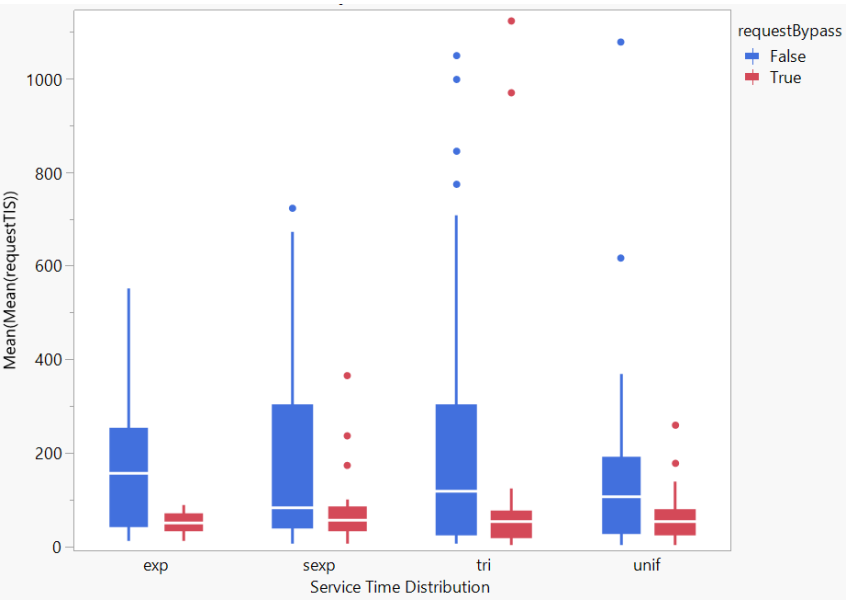


Figure 13. Mean `requestTIS` by Service Time Distribution, `requestBypass`

Another important interaction, shown in Figure 14, compares the mean `requestTIS` per design point by service time distribution for one, two, and three `maxServers` (blue, red, and green respectively) from the grouped dataset. There are only two overall generalizations we observe. First, the range of mean `requestTIS` when `maxServers=1` is larger for all service time distributions. Second, the patterns of the medians for mean `requestTIS` when considering `maxServers` differs by service time distribution. When the exponential distribution is used, there is hardly any difference in medians. However, having `maxServers=3` is not always better, always worse, or always similar to having `maxServers=2`.

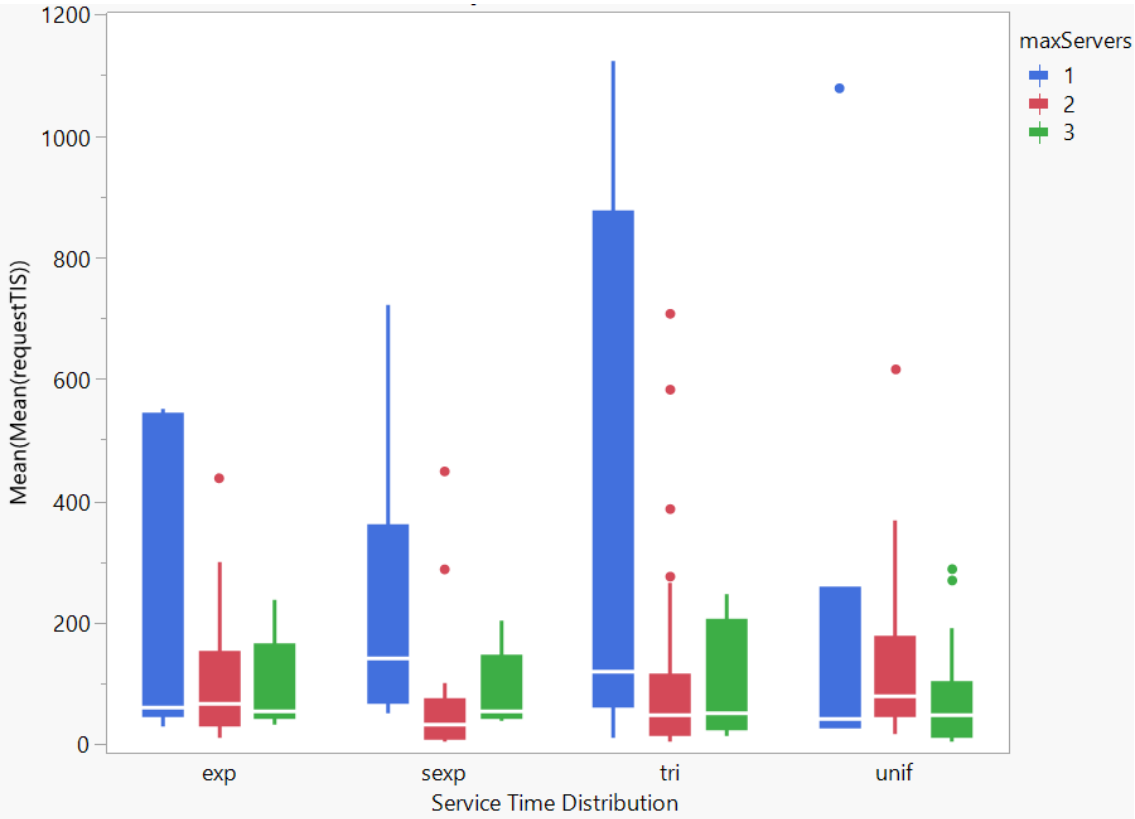


Figure 14. Mean `requestTIS` by Service Time Distribution, `maxServers`

The observations from the interactions with service time distribution factor illustrate significant differences in the behavior of the mean `requestTIS`. If one approval authority is removed from the process, then `requestTIS` might not decrease as much as anticipated depending on the true service time distribution. Changing the number of people in each unit to approve requests, representative of changing `maxServers`, is also likely to lower `requestTIS`. However, it may not yield as significant of a decrease as would be predicted without knowing the service time distribution. Even if the medians for `requestTIS` followed a predictable pattern in Figures 13 and 14, there is substantial variability among these different service time distributions when considering both `requestBypass` and `maxServers`.

C. STEPWISE REGRESSION METAMODEL

Using linear regression to fit the farmed data allows us to observe important factors that influence `requestTIS`, the dependent variable and our MOE. This study uses single, crossed, and polynomial terms with a minimum Bayesian Information Criterion (BIC) stepped forwards for all stepwise regression models. In other words, all main effects, two-way interactions, and quadratic effects are potential terms for inclusion in the stepwise model. Initially, stepwise regression of the dataset grouped by design point yielded a model with 23 terms and an R-squared statistic of 0.981. We made the decision to pare this down to a more parsimonious, interpretable model with 12 terms and an R-squared statistic of 0.975. The absolute values of the t-statistics in Figure 15 are mostly larger than 2, indicating that most of the listed terms are significant. By convention, the main effect for a factor retained in the metamodel if the factor appears in any significant quadratic or interaction terms.

The four most influential factors on `requestTIS`, as listed in Figure 16, are `maxServers`, `availability`, `arrivalRate`, and `svcTimeMean`. The request `arrivalRate` and `svcTimeMean` both have a direct relationship with `requestTIS`; as more requests arrive, the time to process each request increases. Several of these factors have an indirect relationship with `requestTIS`. This infers that increasing the `availability` and `maxServers` will decrease `requestTIS`. We observe that, while `incrementRange` does not have a strong direct effect on the `requestTIS`, it does have an effect when interacting with `arrivalRate`.

Term	Estimate	Std Error	t Ratio	Prob>|t|
svcTimeMean	120.09353	3.538602	33.94	<.0001*
arrivalRate	226.00693	7.588015	29.78	<.0001*
maxServers	-145.4305	5.921105	-24.56	<.0001*
(arrivalRate-1.07151)*(maxServers-2.03093)	-173.9232	12.83432	-13.55	<.0001*
(arrivalRate-1.07151)*(svcTimeMean-2.00551)	76.569805	7.820143	9.79	<.0001*
(svcTimeMean-2.00551)*(maxServers-2.03093)	-55.9857	6.667669	-8.40	<.0001*
(maxServers-2.03093)*(maxServers-2.03093)	76.774389	9.838569	7.80	<.0001*
availability	-107.9728	21.63652	-4.99	<.0001*
(arrivalRate-1.07151)*(availability-0.57495)	-98.49421	42.32878	-2.33	0.0224*
(arrivalRate-1.07151)*(incrementRange-25.5052)	1.0833591	0.500909	2.16	0.0334*
(maxServers-2.03093)*(availability-0.57495)	60.106365	33.49147	1.79	0.0763
incrementRange	-0.024549	0.293231	-0.08	0.9335

Figure 15. Important Terms from Final Regression Model

However, the interactions further down the list in Figure 15 indicate that the benefits of changing a single factor's value is sometimes limited. Once `arrivalRate>1`, then the effect of raising the `maxServers>2`, `svcTimeMean>2`, `availability>0.57`, or `incrementRange>0.25` is negligible. This is due to the flattening effect of increasing `maxServers>2` and maybe partially due to the very large range of `arrivalRate`. Figure 16 displays the interaction plots of the single terms in the stepwise regression metamodel, with the factor values set corresponding to the interaction terms in Figure 15.

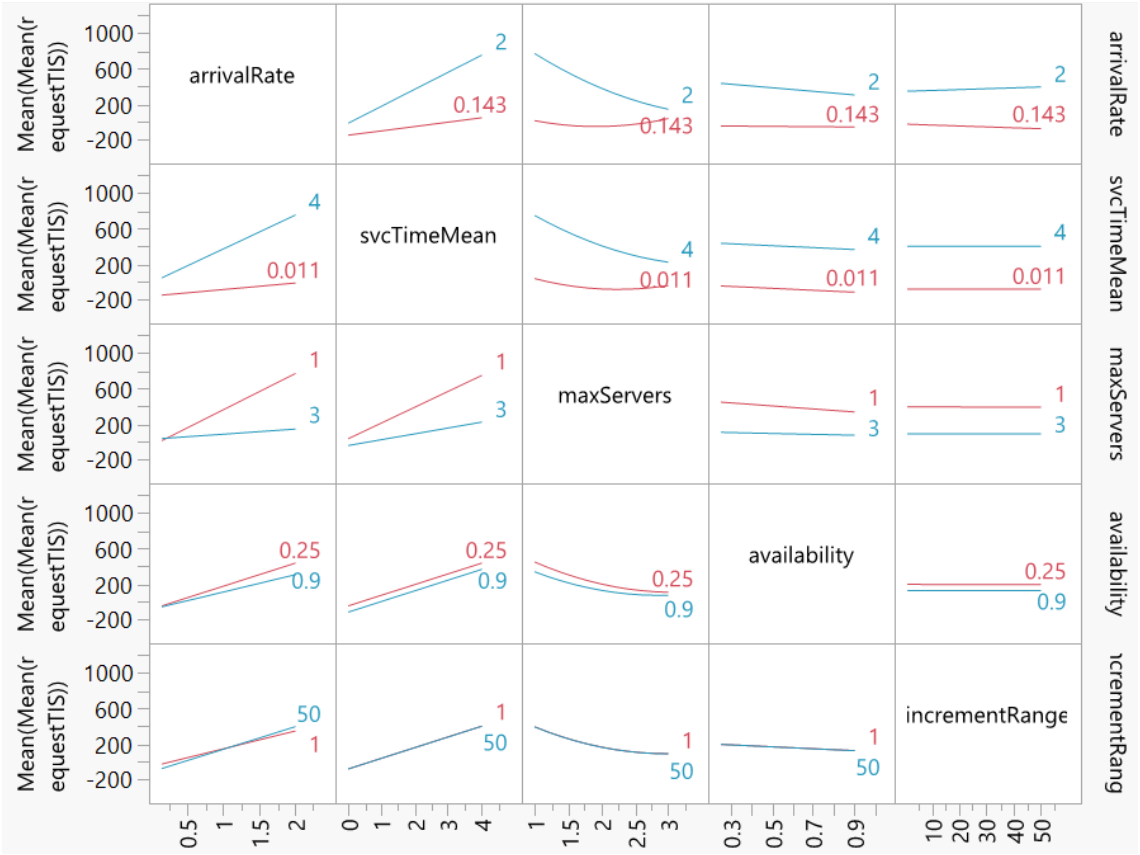


Figure 16. Interaction Plot

When we use these important terms and consider them by design point, we have an extremely high goodness-of-fit with a high probability of predicting the `requestTIS` within 41 days, as displayed in Figure 17. The root mean square error (RMSE) seems high, but this translates to 3.8%. It is a narrow range when the anecdotal estimated time to complete the request is six months and the maximum time is over 1,000 days. The metamodel is likely to change, and its associated RMSE is likely to reduce substantially, if we remove some of most unfavorable and more unrealistic factor configurations from the grouped dataset; here, all the factor configurations are considered.

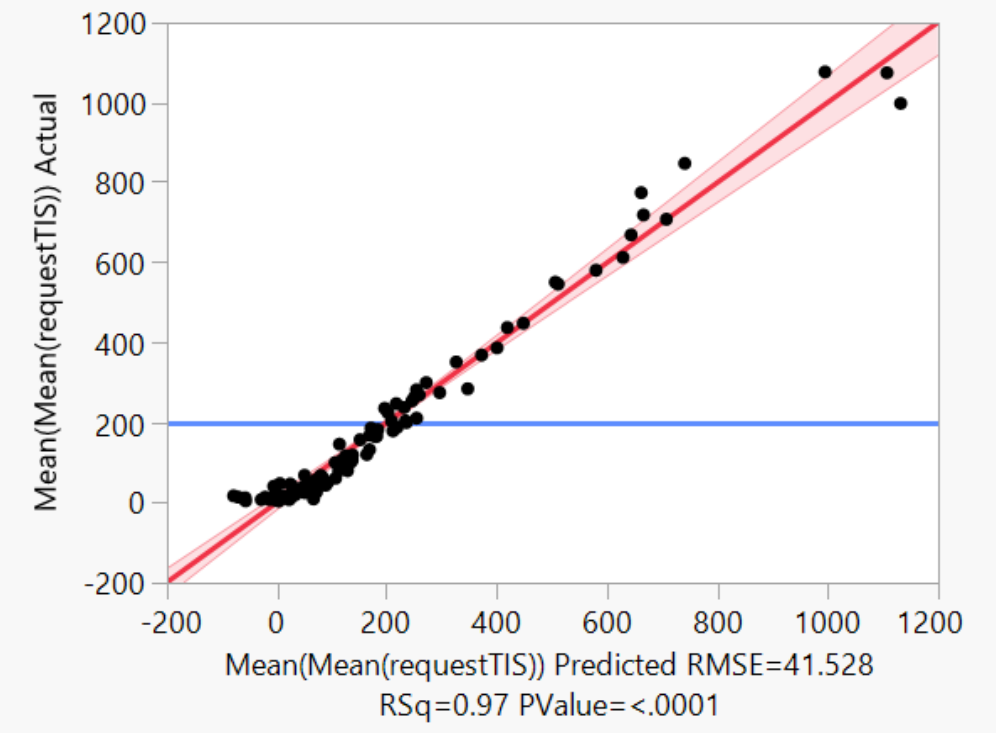


Figure 17. Mean `requestTIS` by Design Point, Actual by Predicted

Essentially, there is an increase in efficiency in this process with two people assigned in each organization to process or approve each request. This team should be available at least 57% of the time and take no more than two full days to process requests to be most effective. Their ability to process requests efficiently decreases once the requests arrive faster than one per day. When the units choose a team and wholistically consider the number of support requests arriving, the number of people assigned to process them, the time it takes them to process the request, and the service members' availability, we can provide a more accurate estimate how long it will take the team to process most of the requests.

D. INCORPORATING PROFICIENCY AS A RANDOM VARIABLE

Proficiency in this model is treated as a random variable with a random learning curve. There are also incorporated assumptions about how requests move through the process even when a unit's proficiency is low, but only after additional time is spent revising the request and gaining a varying level of proficiency. Adding this variability is a

necessary step towards incorporating true human behavior into models for a more realistic approach to processes that involve people at all levels (Connable et al. 2018).

Even though proficiency is not a specifically varied factor, it significantly affects the model indirectly though the `retainRate`, `incrementRange`, and `incrementMin` factors. Figure 18 depicts a partition tree using these three factors to predict the mean `requestTIS` across design point replications. A tree with 10 splits yields an R-square value of 0.39.

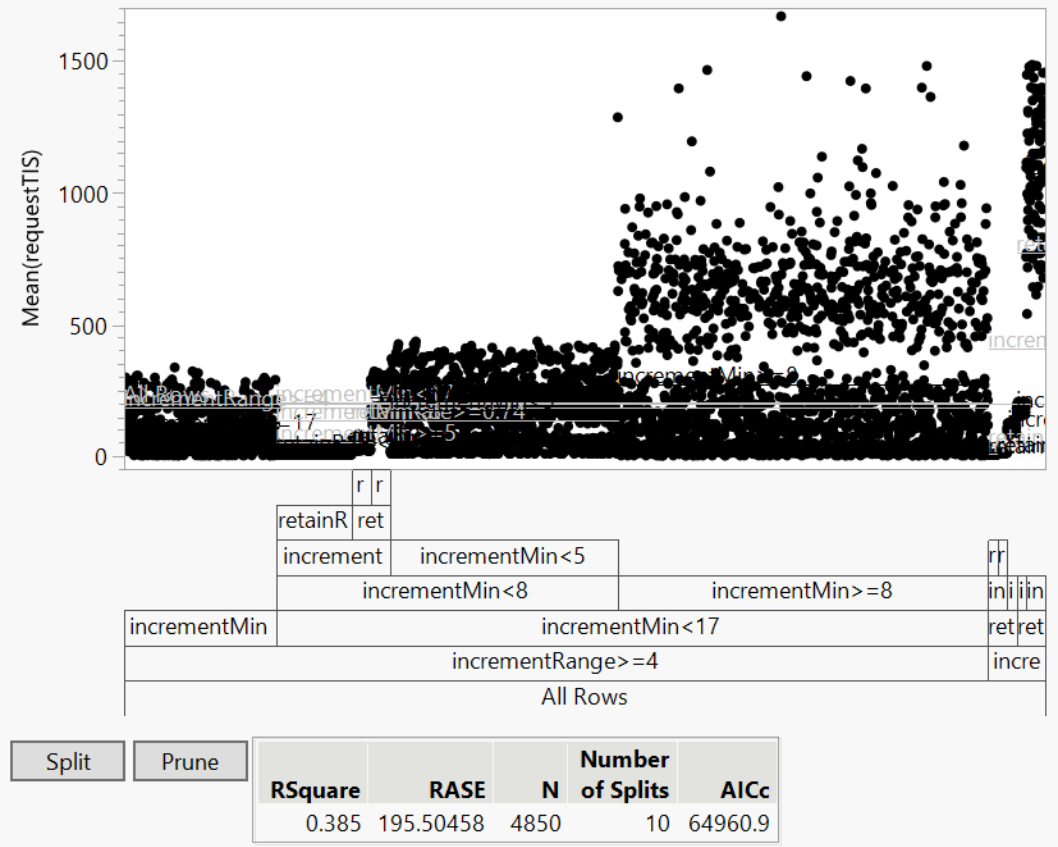


Figure 18. Mean `requestTIS` by Design Point, Proficiency Factors

We observe from Figure 19 that the first three important factor splits are `incrementRange≥4`, `retainRate≥0.74`, and `incrementMin≥17`. If we only use the factors that affect proficiency to predict `requestTIS` and dismiss all others, which are based on reasonable estimations and not data collection, then the R-square

statistic is 0.39. This shows nearly 40% of the variability in the time to process requests can be explained by these three factors, all of which are related to the random proficiency and learning curve behavior. When we closely examine the partition tree splits in Figure 19, we find practical importance in the differences that can be achieved in the predicted values. For instance, we can lower the mean `requestTIS` to 95 days (from the 200 days for the entire farmed data) if fast learners with medium to high proficiency are tasked to process requests. If these appointed individuals are slower learners, then we can lower the mean `requestTIS` to 42 days as long as their team maintains an overall retention rate of 74%. High initial proficiency values or fast learning curves may also represent the difference between formalized, specific training on processing requests as compared to incremental on-the-job training.

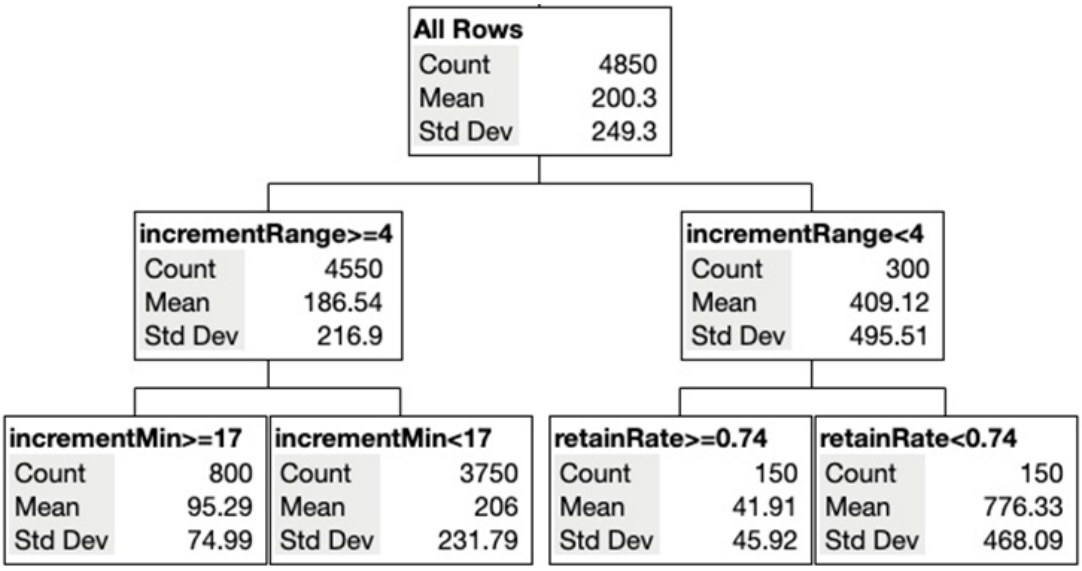


Figure 19. First Three Levels of 10-Split Partition Tree from Figure 18

Adding proficiency as a variable in this model has significant impact on developing a more realistic model to provide more thorough, thoughtful analysis. The human behavior of collective proficiency and learning is extremely relevant, given that our inputs of `arrivalRate` and `svcTimeMean` have a large range based on what seems reasonable, and not from collected data. Being able to first consider these human behaviors, model

them appropriately and realistically into simulations, and observe how they affect data and MOEs is critical to providing insights into the OCO support request process. More importantly, this exercise of modeling the impact of human behavior is important for general business processes, or any process or task where humans are the main connectors that bind the flow of information, products, or services.

V. CONCLUSION

This thesis first introduces the problem of processing tactical cyber requests and proposes simulation with data farming for efficiency testing. Chapter II discusses U.S. Army Cyber operations development and the importance of modeling human factors in general business processes. Chapter III outlines the simulation model structure, development, and application of modeling human processes. Chapter IV discusses our analysis of the farmed data with observations and additional experimentation. In this final chapter, we present recommendations based on the results described in Chapter IV and provides guidance on potential future research.

A. COLLECT DATA FOR SERVICE TIMES

There are significant differences in mean `requestTIS` times when comparing these average times across service times with exponential, shifted exponential, uniform, and triangular distributions. These differences are important to consider for future runs and adaptations of this request process model. Using only one distribution for further experimentation may not give a realistic depiction of the intended process.

We recommend the implementation of a focused effort to collect data about the service times in the organizations involved in this request process. Collecting data about service times drives the ability to use a more precise range of input for `svcTimeMean`, as well as to determine service time distribution for each organization and across the entire process. Experiments may then be conducted with service time distributions which more closely represent the actual times for organizations to process, revise, and approve CERFs.

B. CHANGES IN THE REQUEST PROCESS TO IMPROVE EFFICIENCY

Even though the variation in the model's service time distributions affect `requestTIS` significantly, we observed some overall practices that can be implemented for improving the OCO support request process. First, it is generally more efficient to have at least two people in each organization to process and approve CERFs. Assigning only

one person to complete this task tends to be highly ineffective and causes much more variability in the overall request process. It may not increase the average time to process a request when three people per organization are assigned to process CERFs, but the processing time is more consistent. Second, removing one authority approval generally makes the overall time to process requests faster. The true impact of bypassing approval authorities on processing efficiency can be tested with further experimentation once we have a known service time distribution, including data on service times.

Both the service times of processing requests and the request arrival rate are very important factors in determining the total time to process a request. They can support processing requests without a decrease in efficiency under these conditions while requests flow through the process of up to one daily. The request load is unmanageable if the requests arrive as quickly two per day. We can estimate the amount of time that a team will take to process requests with relative accuracy when we consider the team staffing, their availability, their current proficiency with processing requests, and their learning rate.

C. FUTURE RESEARCH

This study explores the process of approving OCO support requests while proposing some changes to increase efficiency. Ultimately, increasing efficiency for request processing provides support more quickly for tactical commanders. There are several additional scenarios and model adaptations to examine to continue to improve efficiency and support to warfighters.

1. Additional Scenarios

In Section A of this chapter, we discussed the need for additional data to support the service time distribution used in the model. If service time data is collected across the organizations that participate in this process, then we highly recommend further experimentation to confirm the findings in this study and expand on them. Collecting any data for the ranges of the varied factors through YAML or using other service time distributions not used in this study's experimentation may also confirm our findings and provide additional insight.

Sensitivity analysis can also be conducted with any of these factors or any of the assumptions made about human behavior from Chapters I, II, and III through the design of experiment and YAML configuration file setup. One specific assumption to modify is allowing requests to be denied based on a level of proficiency, randomness, or another factor not mentioned. Experimentation using these scenarios further tests the robustness of and reduces uncertainty in this simulation model.

2. Model Adaptations

This study uses a formally outlined process with a published workflow. However, there are typically parts of a request process that include sending or revising requests through an informal network. One example of existing informal relationships that can affect the OCO request process is the addition of the DIV S6/CEMA organization. This group has a peer-to-peer relationship with the DIV S3 (*Q3*) and a supervisor-mentor relationship with the BDE S6/CEMA (*Q1* as the original requester). This informal relationship can help reduce revisions and raise proficiency through mentorship with Q1; it can also raise awareness for *Q3* through the peer relationship, producing a similar effect of bypassing the approval authority of *Q2* by stressing the urgency of the request. This informal network can be added within the model for additional insight and understanding the effectiveness of using an informal network for request processing.

3. Implications for Future Study

The OCO support request process is the only process modeled in this study. However, the DCO support request process is more complex. This model can be adapted to reflect the workflow of that process to provide specific insight or compare general insights across both support request processes to find commonality if it exists. In fact, the U.S. Army has many similar workflow processes ranging from logistic support requests to processing orders through a chain of command and chain of support that can benefit from efficiency testing through simulation. This simulation model, specifically with the incorporation of human behavior into quantifiable parameters and use of a designed experiment to produce reasonable, farmed data, can be adapted for any workflow process or task to save time for commanders and support warfighters.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX: AVAILABLE FILES FOR USE

The files used in this study are available by request, or at the links below for use:
https://gitlab.nps.edu/anna.hughes/OCO_support_request
https://github.com/annacathughes1/support_request

1. simplekit.py
2. tandem.py
3. OCOrequest.py
4. config.yaml
5. nohls.txt
6. OCO_Request_Process_Event_Graph.png

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Baird S and Özler B (2012) Examining the reliability of self-reported data on school participation. J. of Development Economics 98(2), https://doi.org/10.1016/j.jdeveco.2011.05.006.

Connable B, McNerney M, Marcellino W, Frank A, Hargrove H, Posard M, Zimmerman S, Lander N, Castillo J, Sladden J (2018) 'Will to Fight': Analyzing, modeling, and simulating the will to fight of military units, RAND Corporation, Santa Monica, CA, https://www.rand.org/pubs/research_reports/RR2341.html.

Department of the Army (2021) Cyberspace Operations and Electromagnetic Warfare. FM 3-12, Washington, DC, https://armypubs.army.mil/epubs/DR_pubs/DR_a/ARN33127-FM_3-12-000-WEB-1.pdf.

Hoffman M (2015) OE conditions for training: A criterion for meeting "objective task evaluation" requirements. Quarter 4. https://www.benning.army.mil/infantry/magazine/issues/2015/Jul-Sept/pdfs/Hoffman-OE%20Conditions_TEXT.pdf

Hollis D (2010) USCYBERCOM: The need for a combatant command versus a subunified command. U.S. Army. Accessed December 19, 2022, https://www.army.mil/article/41585/uscybercom_the_need_for_a_combatant_command_versus_a_subunified_command.

Horne G and Seichter S (2010). Data farming in support of NATO operations - methodology and proof-of-concept. Proceedings of the Winter Simulation Conference 2014, Savannah, GA. DOI 10.1109/WSC.2014.7020079.

Joint Chiefs of Staff (2018) Cyberspace operations. JP 3-12, Washington, DC. https://www.jcs.mil/Portals/36/Documents/Doctrine/pubs/jp3_12.pdf.

Kallfass D and Schlaak T (2012). NATO MSG-088 case study results to demonstrate the benefit of using data farming for military decision support. Proceedings of the 2012 Winter Simulation Conference (WSC), Berlin, Germany, DOI 10.1109/WSC.2012.6465132.

Leibowitz N, Baum B, Enden G, Karniel A (2010). The exponential learning equation as a function of successful trials results in sigmoid performance. *J. of Mathematical Psychology* 54(3): 338–340. DOI 10.1016/j.jmp.2010.01.006

Merrett F (2006) Reflections on the Hawthorne Effect. Educational Psychology 26(1), DOI 10.1080/0443410500341080.

Sanchez PJ (2007) Fundamentals of simulation modeling. Department of Operations Research, Naval Postgraduate School, Monterey, CA.

Sanchez PJ and Sanchez SM (2023) Tandem.py module with simplekit module imported via email. January 27.

Sanchez SM and Sanchez P (2020) Robustness revisited: Simulation optimization viewed through a different lens. Proceedings of the 2020 Winter Simulation Conference, eds. K.-H. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing, Piscataway, NJ: IEEE. https://informs-sim.org/wsc20papers/005.pdf

Sanchez SM (2018) Data farming: Better data, not just big data. Proceedings of the 2018 Winter Simulation Conference, eds. M. Rabe, A. A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, Piscataway, NJ: IEEE, 237–251. https://www.informs-sim.org/wsc18papers/includes/files/035.pdf

Schoeni, DE (2017). Still too slow for cyber warfare: why extension of the rapid acquisition authority and the special emergency procurement authority to cyber are half measures. *Public Contract Law Journal*, 46(4), https://www.jstor.org/stable/26419554

Senft M (2022) Cyber mission planning: Targeting, November 2, 2022, Department of Information Science, Naval Postgraduate School, Monterey, CA.

Suarez J (1992) Three experts on quality management: Philip B. Crosby, W. Edwards Deming, Joseph M. Juran. TQLO Publication No. 92–02, Total Quality Leadership Office, Department of the Navy, Arlington, VA, https://apps.dtic.mil/sti/pdfs/ADA256399.pdf.

White S (2019) Subcultural influence on military innovation: the development of U.S. military cyber doctrine. Doctoral dissertation, Harvard University, Graduate School of Arts & Sciences, Cambridge, MA, http://nrs.harvard.edu/urn-3:HUL.InstRepos:42013038.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California