ENABLING AIR FORCE SATELLITE GROUND SYSTEM AUTOMATION THROUGH SOFTWARE ENGINEERING

THESIS

Michael J. Bentley, 2nd Lieutenant, USAF

AFIT-ENG-MS-17-M-006

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS-17-M-006

ENABLING AIR FORCE SATELLITE GROUND SYSTEM AUTOMATION THROUGH SOFTWARE ENGINEERING

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

Michael J. Bentley, BS

2nd Lieutenant, USAF

March 2017

AFIT-ENG-MS-17-M-006

ENABLING AIR FORCE SATELLITE GROUND SYSTEM AUTOMATION THROUGH SOFTWARE ENGINEERING

Michael J. Bentley, BS

2nd Lieutenant, USAF

Committee Membership:

Maj A. C. Lin, PhD
Chair

Lt Col J. M. Pecarina, PhD
Member

Dr. D. D. Hodson, PhD
Member

Abstract

US Air Force satellite ground stations require significant manpower to operate due to their fragmented legacy architectures. To improve operating efficiencies, the Air Force seeks to incorporate automation into routine satellite operations. Interaction with autonomous systems includes not only daily operations, but also the development, maintainability, and the extensibility of such systems. This thesis researches challenges to Air Force satellite automation: 1) existing architecture of legacy systems, 2) space segment diversity, and 3) unclear definition and scoping of the term, "automation." Using a qualitative case study approach, we survey comparable non-satellite operation domains (Industrial Control Automation and Software Testing) that have successfully integrated automation and other satellite operation enterprises (NASA Goddard, Naval Research Laboratory, European Ground Station National Institute for Space Research in Brazil) to identify common themes and best practices. From this insight, we recommend that future satellite operation ground stations encourage the use of layered architectures, abstract satellite operation processes, and integrate simulators in future systems as concrete implementations of this common operating platform. Further elaborating on the value of these recommendations, this thesis researches the benefits of process mining in satellite operations. This research is conducted on FalconSAT-3 and PropCube 1 and 3. The process mining analysis discovered that a highly structured Concept of Operations (CONOPS) is required in order to gain significant benefits from process mining.

Acknowledgments

I would like to offer my heartfelt thanks to my advisor, Major Alan Lin for the many hours he spent meeting with me on a weekly basis for a year. Thank you for helping me organize site visits and for providing direction along the way. I would also like to thank the people who worked hard to provide me with data for my research; Captain Brian Kester at the United States Air Force Academy and Mr. Giovanni Minelli at the Naval Postgraduate School. Also the excellent people at Kirtland, AFB, specifically, 1st Lieutenant Daniel Skaggs and Captain Stephen Yoshimura. Finally, I would like to offer a special thanks to my wonderful wife who put up with all of the travel, the late nights, and weekends. For always being there for me, and reassuring me whenever the going got tough, thank you.

Michael J. Bentley

Table of Contents

Page

List of Figures

Page

List of Tables

Page

ENABLING AIR FORCE SATELLITE GROUND SYSTEM AUTOMATION THROUGH SOFTWARE ENGINEERING

I. Introduction

1.1 General Issue

Military satellite operations have been slow to adopt automation into systems and processes. Automation provides reliability, efficiency, safety, and rapid response capabilities as demonstrated in other domains such as Industrial Control Systems, Software Test and Evaluation, and Intelligent Transportation Systems. While some of the reluctance to adopt automation stems from cultural and organizational concerns, the United States Air Force (USAF) does recognize the advantages, given examples from the civil and commercial space industry. This research identifies and addresses technical and methodological obstacles to satellite ground station automation in the USAF.

1.2 Problem Statement

The USAF has lagged behind the trends in automation for many years. Organizations like the USAF Scientific Advisory Board have made the point for the past sixteen years that the USAF desperately needs to adopt faster acquisitions processes and leverage the work of commercial and other governmental organizations [1]. This point has been reiterated by multiple Government Accountability Office (GAO) reports over the past decade [2]. In order improve this situation, the USAF must design the next generation ground system with the future of automation in mind. Three overarching conditions contribute to the difficulty of adopting automation into military satellite operations.

First, most USAF ground stations were not built with automation in mind [2]. Adding automation to an existing ground system is expensive. A foundational principle of software engineering is that adding requirements after a design is complete will be more difficult and costly than considering and planning for them earlier in the development cycle. This not only reinforces the difficulty of integrating automation with disparate, legacy systems, but also the importance of developing the next generation of ground systems with automation and future advances in mind.

Second, each satellite is unique. Possibly the most significant difference is due to each satellite's orbit. Geostationary or geosynchronous orbits have very different mission requirements than highly elliptical or low-Earth orbits. Additionally, satellite buses are manufactured by a wide range of vendors using a combination of custom and standard parts with additional modifications [3]. The larger the constellation of similar or identical satellites, the greater the return on investment (ROI) from automation efforts will be. For example, Planet Labs' constellation of approximately 150 imaging satellites, with 15-30 satellites of the same model in the same orbit, can achieve more ROI from automation than the USAF which has fewer, larger satellites, in many different orbits [4]. The diversity in the space segment complicates the automation process and achieving ROI requires finding ways to maximize code reuse.

Third, automation is frequently used in vision statements and planning documents [5], but there is no firm definition of what, "ground station automation" actually means. The IEEE Robotics and Automation Society defines automation as the "use of automated methods in various application to improve performance and productivity" [6]. In the last fifteen years, ground station automation has primarily been constrained to scripting

satellite passes. Some ground stations have extended automation to handling the process of software set up and tear-down for a satellite pass, while others still require manual set up before a scripted pass occurs. On the far end of the automation spectrum, a "lights out" ground station requires no human intervention for routine operations. However, even these systems generally still require human intervention when performing planning and scheduling in advance, though greater automation in planning and scheduling has become more common in recent years [7]. A clear definition of USAF automation is crucial in order to determine whether automated satellite operations met its goals.

The objective of this research is to identify technical approaches that address these challenges to enable automation and reap its benefits. To identify solutions, this thesis poses the following research investigative questions.

1.3 Research/Investigative Questions

The following research question was identified through literature review as well as site visits to the USAF's Multi-Mission Satellite Operations Center (MMSOC) at Kirtland AFB, NM.

1.3.1 Research Question: How can the USAF make future satellite automation more capable and less expensive?

"More capable" means automation will be able to cope with a wider range of circumstances and generally require less human intervention at all phases. The ultimate goal of automation would be a system that requires no human intervention except for the use of force, as will be explained later on. "Less expensive" means that the benefits of automation (reliability, fewer operators, and greater capability) must outweigh the costs

of developing, implementing and maintaining that automation. This question will be answered through extensive examination of the following investigative questions.

1.3.2 Investigative Questions:

- What are the current trends in satellite operations (SATOPS) automation?
- Is the USAF keeping pace with these trends?
- What are the automation trends in mature, non-space domains that might be applicable?
- How do we know if a mature domain is applicable?
- How can we make automation less expensive and more flexible?
- How do we determine Return on Investment (ROI) for automation?
- What gives us the highest ROI to automate?
- What are the limits of automation?
- What are routine versus non-routine SATOPS?

1.4 Methodology

The methodology applied to this research can be broken down into two distinct phases. The first phase is a case study analysis of automation in existing ground stations as well as automation in several non-space domains. The focus of this case study, using a software engineering principles and design checklist, was to identify trends and gaps [8]. Figure 1 shows the research outline followed by this thesis.

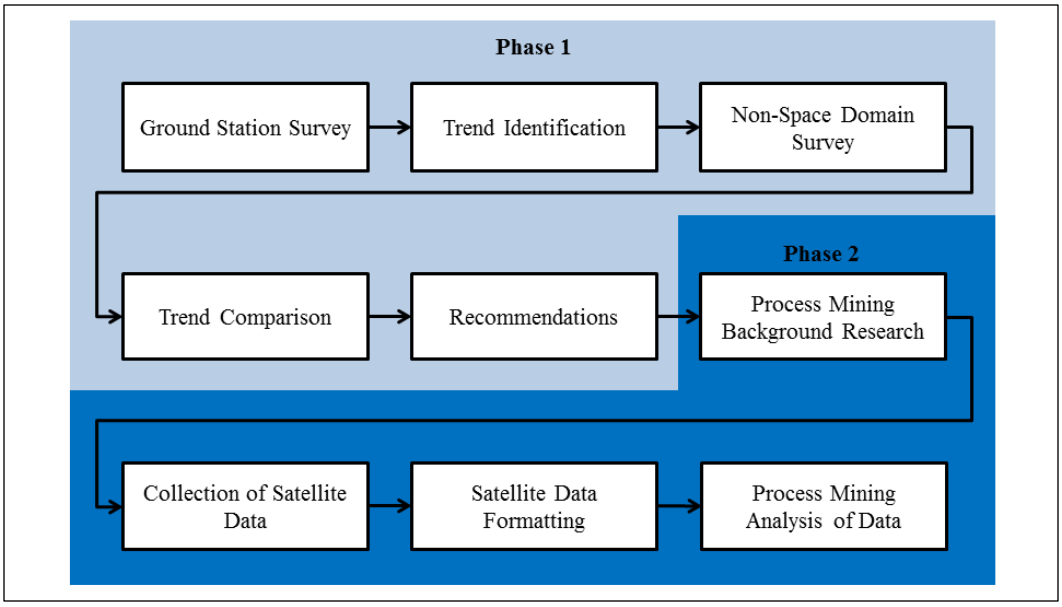


Figure 1. Two-phase research outline

Phase 1 addresses an identified research gap shown in Figure 2. The figure identifies multiple instances of case study analyses in the space domain and in the non-space software engineering domain.

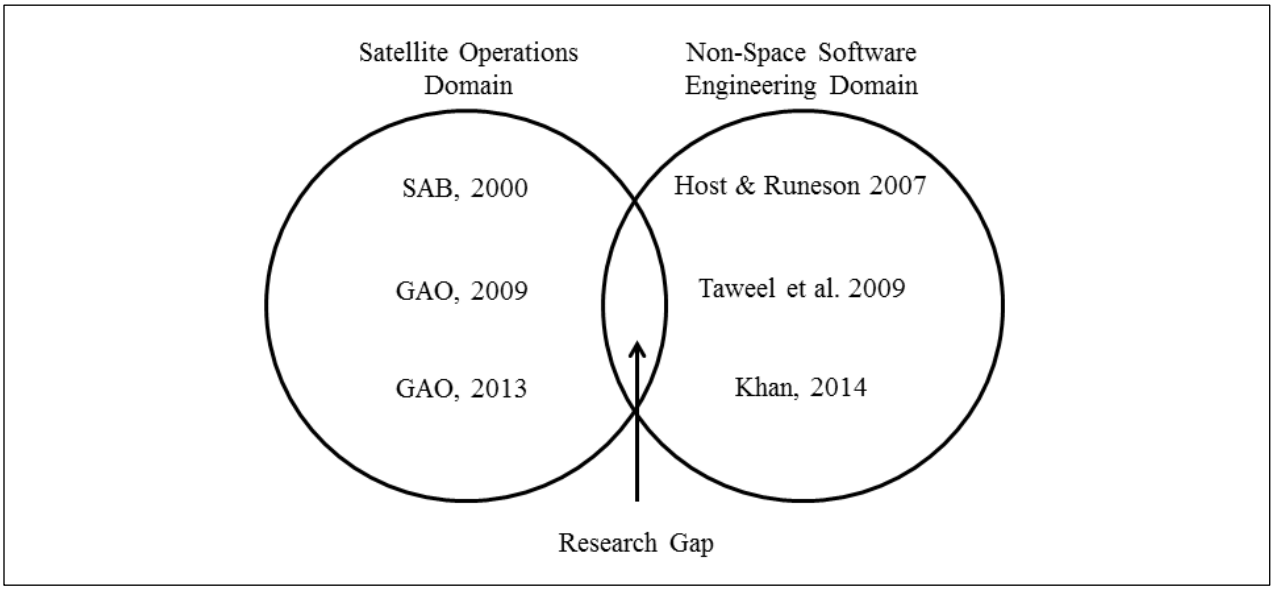


Figure 2. Case study research in space and software engineering domains

Previous works have stayed squarely in one category or the other. This research reinforced trends supported by case study analysis within the SATOPS domain. It also discovered new trends from three non-space domains that have direct applicability to the problem of automation and flexibility in SATOPS.

The second methodological phase focused on recommendation 3 because this recommendation was found to be most difficult for subject matter experts (SMEs) to understand. Even when the method of increasing abstraction in the design of automation is fully understood, SMEs in the SATOPS domain did not intuitively recognize the value it would provide. Therefore, this experiment demonstrates the value of applying the recommendation of *increased abstraction* identified in phase one. The experiment uses process mining techniques to identify highest ROI activities for automation as well as to develop a pattern of life model for multiple satellites. However, without the implementation of *increased abstraction* the process would be largely manual and inefficient. Thus, the experiment demonstrates potential value to be gained by using greater abstraction.

1.5 Assumptions/Limitations

Because much of the information regarding US government ground stations was gathered through interviews and site visits, we must assume honest and complete answers were given to questions about their architecture. We believe this assumption is valid because open-source documentation largely agrees with the representation of these ground stations provided by operators and engineers.

A second assumption is that the range of ground stations chosen for this case study analysis is large enough to infer trends in SATOPS. This assumption is valid to the extent that these recommendations identify differences between trends in large, government-controlled, ground stations and non-space domains. This research is not trying to provide an exhaustive survey of modern ground-stations. Notably absent are commercial ground stations as well as Russian, Chinese, and Indian ground systems, to name a few.

This research was constrained by several significant limitations. The first limitation was the difference in research sources used for non-space and space domains. This limitation was partially based on the fact that organizational connections facilitated site visits to various ground stations, while non-space domains were studied exclusively through scholarly publications. It is difficult to predict what impact this disparity may have produced in recommendations.

A second limitation is the range of ground stations and non-space domains considered. Given limited time and resources, it was not practical to visit additional ground stations or identify more non-space domains to research in-depth. Directly connected with this limitation is that access to some of the most cutting-edge ground stations in the world is heavily restricted. This could challenge the assumption that the ground stations considered here are representative of large governmental ground systems. However, as mentioned under assumptions, this was mitigated by using additional ground stations from non-US government institutions. As these were found to largely coincide with trends in the US, they reinforced and even expanded the trends found in US ground stations.

The third limitation was the high degree of difficulty in acquiring real-world satellite data. This applies to the second phase of analysis. Many organizations are wary of providing in-depth data from operational satellites due to operational security and proprietary information concerns. As a result, experimentation used data for three small satellites collected from academic institutions. While not as big nor complex as DoD satellites, they provide a basis for proof-of-concept. Given the infancy of this research, limited data sets, simulated data, and extensive cube satellite data were compiled to produce an initial analysis. This analysis would be greatly strengthened by acquiring large data sets from more advanced satellites.

1.6 Implications

This research has two distinct categories of implications. The first implication of this research is that case study analysis using non-space domains can identify novel solutions to SATOPS problems. This has been noticed in other areas of the space domain, such as optics, but seems to be under-utilized in the software engineering sector. The second category of implications is that through the recommendations presented here, SATOPS is on the verge of moving from 20th century concepts of automation into the machine learning era. Other domains have taken the concept of automation much farther than space. Through loosely-coupled, layered architectures, increased abstraction, and integrated simulation, the door will be opened to advanced techniques in artificial intelligence and human-machine teaming. In the short term, the implication of these recommendations is that the USAF has an opportunity to move ahead of state-of-the-art

and simplify the process of integrating diverse satellite operations into the same architecture.

1.7 Document Overview

Chapter II presents the background research conducted in space and non-space domains. This research answers many of the investigative questions presented in Chapter I. Chapter III presents the methodology used in studying the background domains. It describes the trends and software engineering principles highlighted by each non-space domain. Chapter IV provides an analysis of the three recommendations produced in Chapter III. Then Recommendation 3—*Increased Abstraction*—becomes the target of the second half of this thesis. First, we present a background of process mining and transfer learning, followed by an experiment demonstrating their value to the space domain. The value of abstraction in enabling these tools is then shown. Chapter V summarizes the recommendations presented here as well as their impact and closes with future work for this research.

II. Background

2.1 Chapter Overview

This section begins with an introduction into the basic process of SATOPS. Next it reviews the different domains and systems used in the case study analysis. We selected non-space domains that exhibited similar operating requirements, such as high maintainability, mission criticality, and reliability, along with autonomy's goals in the domain of reducing repetitive tasks. We focused on non-USAF systems that also have high space segment diversity.

2.2 Introduction to SATOPS

The basic process of operating a satellite varies depending on the type of orbit a satellite is in. Satellites in geosynchronous or geostationary orbits, referred to as *GEO,* generally remain in constant contact with their ground station because they move very little, relative to the ground. Thus, they can maintain persistent data links. Satellite's in lower orbits—for example medium Earth orbit, *MEO* or low Earth orbit, *LEO*—have a pass window for uplink and downlink. This window is the period of the satellite's orbit during which it has line of site to a ground station. While the satellite is in contact, commands may be sent and responses can be recorded. The most important activity to perform is to tell the satellite to transmit all of its telemetry information. This is generally known as the *state of health.* With this information, the operator, or an automation script can make decisions about what activities need to be performed by the satellite immediately or later, when the satellite is out of contact with the ground. In the majority of government ground stations, an operator is required to directly command the

satellite by sending a series of commands written down in a Concept of Operations (CONOPS). This is essentially a checklist that the operator follows on every pass. If there is a problem with the satellite, known as an *anomaly*, an engineer is always on call to perform troubleshooting. Automation scripting is when the sequence of commands the operator must perform is prearranged with some logic to determine actions based on telemetry from the satellite. Figure 3 demonstrates this standard process.

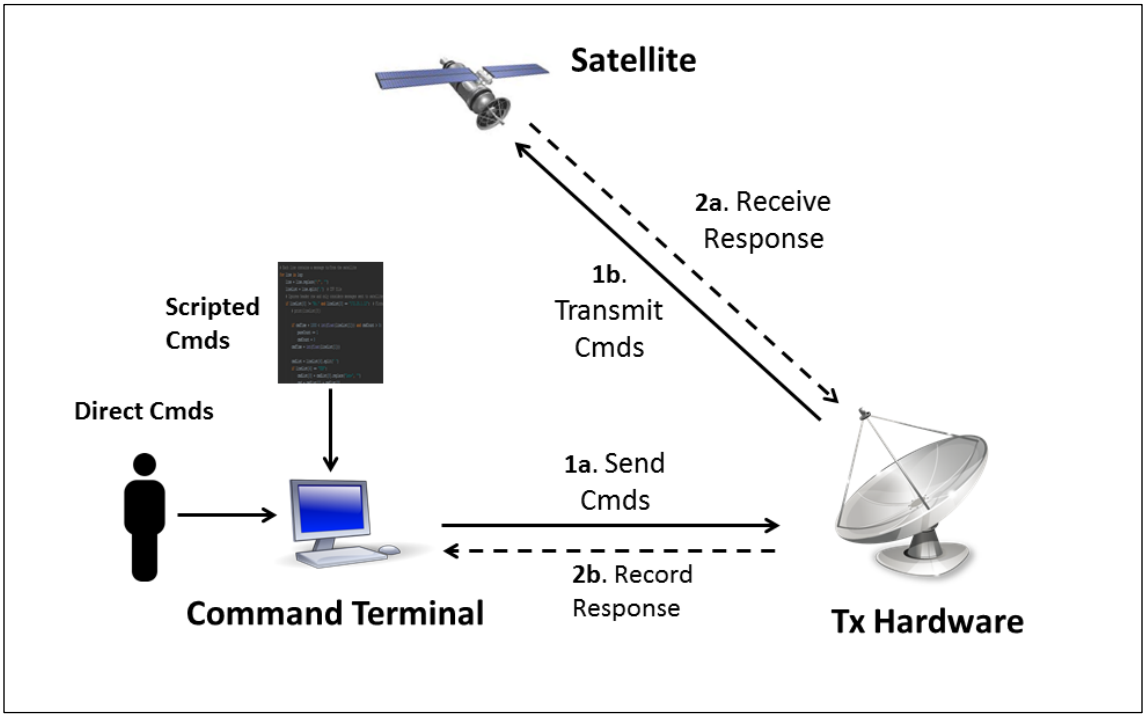


Figure 3. Generic SATOPS overview

Advanced ground stations, like the Naval Research Laboratory's Blossom Point Tracking Facility have extended their automation to include all of the steps required to set up a command terminal for a pre-scheduled pass and select appropriate automation scripts. Their ground station is considered, "lights out" as mentioned in section 1.2 because no humans are required beyond the planning and scheduling phase. This is of course only

the case if nothing goes wrong with the satellite. In the event of an anomaly, an engineer must still be brought in to resolve the issue.

2.3 Relevant Research

Research was conducted in two specific categories. The first category includes several non-space domains. These domains will be considered in both here and in greater detail in Chapter III. The second category includes 4 examples from the SATOPS domain, including two US government ground systems and two foreign government organization ground systems.

2.3.1 Non-Space Domains

In order to better understand the principles that support ground station automation we consider several mature domains with similar system performance requirements: 1) Industrial Control Systems, 2) Test and Evaluation, and 3) Intelligent Transportation Systems. The primary method of data collection for each non-space domain was literature review.

2.3.1.1 Industrial Control Systems (ICS)

In terms of performance requirements, industrial automation shares many similarities to satellite ground stations – they cannot tolerate failure (i.e. being "down"), yet need to offer a high degree of flexibility for updates and modifications as processes change. Software engineering has advanced the design of ICS significantly in software

lifecycle efficiency and cost savings [9]. Flexibility is crucial in industrial automation because, like space, multiple engineering domains overlap; for example, mechanical engineering, electrical engineering, and software engineering [10]. Space systems further incorporate aerospace engineering. Developments and advancements in any one of these domains may cause critical design changes. As a result, software architectures must be flexible in order to stay current with rapidly changing technology. Industrial automation also leverages and recognizes the importance of modeling large-scale automation projects [10]. Modeling helps define functional requirements and how the software (i.e., components) should behave and interact with each other. Developed software must pass rigorous test criteria to ensure it performs as desired. These requirements connect directly to the next domain.

2.3.1.2 Software Test and Evaluation (T&E)

Manually testing software, even against available models, is laborious and time consuming. Therefore, the T&E domain has invested in test automation. As a well-established domain, T&E is relevant to the space domain for several main reasons. First, future ground station automation should be using test automation directly in the software development lifecycle. In fact, as of 2016 NASA has published a paper on the importance of automated model based testing in designing complex software systems [11]. This reinforces the underlying principle of modeling that supports all of the recommendations presented in this thesis. Other researchers have published similar papers in 2016, reinforcing the importance of T&E automation in the space domain [12].

The second reason T&E is relevant to the space domain is because the principles that enable test automation are transferrable to other systems pursuing greater automation. Two of the most important principles are abstraction and dependency minimization [13]. Test automation must find ways to present a wide range of equivalent or similar data to a system. This is a challenging problem that can use abstraction in order to categorize inputs by their commonality. The principal of abstraction is crucial to generalizing ground station automation. Minimizing dependencies is essential for flexibility and automation because it limits the second and third-order effects of modifying or interacting with components. Minimizing dependencies is frequently done through encapsulation and introducing standardized interfaces [14], which is essential to enabling automation to interact with multiple components in a similar manner. Finally, the T&E domain also reinforces the importance of designing systems with automation in mind and prioritizing what functions are automated based on highest return on investment (ROI), [15].

2.3.1.3 Intelligent Transportation Systems (ITS)

The ITS domain is the newest of the three non-space domains considered here. Despite its young age, it enjoys significant publicity and research. While satellite ground control automation has the potential to make satellite systems less expensive and more reliable, ITS could transform the world in the not-so-distant future. ITS makes for an excellent case study because it shares many requirements and operating characteristics of the space domain. Safety and reliability are the most important requirements of both

domains. The satellite control domain is concerned with preserving multi-billion dollar satellites, while the ITS domain is concerned with the safety of transporting millions of people. In this respect, ITS is one of the few domains with potentially higher stakes than controlling national security space assets. The ITS and SATOPS domains are also concerned with similar questions. For example, ITS asks how much intelligence to build into vehicles versus how much to rely upon the intelligent transportation infrastructure to manage the vehicles. This question is currently being addressed in the satellite domain. While automation moves forward in ground stations, it is also being expanded in satellites themselves. Finally, the ITS domain faces the same specific challenge that the SATOPS domain is currently seeking to overcome. That is, a wide range of vendors producing different solutions that must somehow integrate into the same intelligent transportation system [16]. Each vehicle manufacturer is pursuing different solutions to automation, and multiple agencies are developing intelligent transportation management systems. In order to integrate these disparate systems, researchers are proposing solutions that have potential value for the SATOPS domain as well.

2.3.2 Non-USAF Satellite Ground Stations

We studied non-USAF space operations ground stations and examined their automation efforts. In addition to their software architecture, we examined how each organization defined the scope and role of automation. We performed literature review, site visits, and interviews in order to determine what requirements the research domains must share in order to be appropriate for case study analysis.

2.3.2.1 NASA Goddard Space Flight Center

NASA's Goddard Space Flight Center (GSFC) operates over fifty spacecraft throughout the solar system. These spacecraft have a wide range of missions. Some study the Earth while others study Earth's moon, Mars, the Sun, and even deep space. At GSFC, engineers have leveraged many of the software design principles described previously. In particular, they have embraced an incremental approach to automating many of their smaller research satellites. Because these projects are either older or less expensive, they have a higher risk tolerance. They also have very tight budgets that do not allow for a one-time "automation" upgrade. Instead, as funding permits, they automate additional functions on the satellite, which in turn frees up money that would have been spent on operating the satellite. Eventually, this produces a well-automated satellite system that continues to run until the satellite reaches end of life, at minimal cost. A key contribution of their automation efforts is the GMSEC message passing interface (MPI). While not directly part of automation, having a standard MPI makes adding system components much simpler because all modules have a common message wrapper, similar to TCP/IP packets. This makes the process of automating the entire ground system easier because an automation script can parse messages from every component with minimal tailoring. The GMSEC MPI is an excellent use of software design principles. Each component already knows what messages will look like, so they can reliably read messages from any other component. GSFC has implemented automation to the lights-out stage in multiple cases, but they purposefully have not

extended their automation to perform anomaly handling—If a system parameter goes red, the system will notify engineers and wait for help. GSFC has not implemented an integrated simulator and relies on tailored automation scripting for each satellite. However, they would still be considered in the "lights out" category for many of their programs.

2.3.2.2 Naval Research Laboratory

The Naval Research Laboratory's (NRL) Neptune architecture takes advantage of loose coupling but also maintains a strict baseline standard that can only be modified by the NRL. Customers can add to this baseline, but cannot alter it themselves. This forces adherence to a standard, but could potentially be an obstacle to rapid development in some cases. They have come farther than almost anyone in applying automation to all aspects of satellite operations. Not only does Neptune automate each contact with a satellite, it also creates a hardware string in real time from a resource pool. This makes the system robust to system failure because all available hardware resources become potential backups and can be replaced in seconds through digital switches. Neptune uses an efficient planning and scheduling algorithm to perform all resource allocation autonomously. They generally plan passes a week in advance and once planned, the system will perform all the steps to create a contact session, establish a hardware string, and fly the pass. They have also implemented scripts to handle known anomalies. For example, in many cases, a simple memory flush, or reboot of a component can resolve known issues. These solutions can be easily scripted and give the system a chance to

correct simple errors before requiring human intervention. Their automation system features three operating modes; monitor, assisted, and autonomous. In monitor mode, the Automated Ground Operations (AGO) system does not perform any actions, but informs the operator what it would do at each step. In assist mode, the AGO performs the actions, but only after the operator approves each step. Finally, in autonomous mode, the AGO performs all actions on its own, but allows the operator to halt the system at any time and take over manual control [17]. Neptune is in many ways the benchmark for government ground stations in the United States. In terms of Telemetry Tracking and Commanding (TT&C), Neptune has many capabilities and features described by senior USAF Space Command leadership. This is an example of how the USAF should take advantage of existing technology, but reach beyond state of the art in developing a system that will be cutting edge for the foreseeable future.

2.3.2.3 European Ground System

The European Ground System—Common Core Initiative (EGS-CC), is one of the most advanced ground stations outside of the US They provide a well-documented example of many best practices in the space domain [18][19]. They are especially relevant to a discussion of USAF automation because EGS-CC is designed to bring together many organizations that were previously connected only loosely. This is not a simple task because every organization has various standards and the ESA has limited direct power over the group. The USAF has total control over the sub-organizations controlling its various ground stations. At times this may be a hindrance, but it should

enable the adoption of good standards once they are chosen. This is something EGS-CC has had to do through diplomacy and in-depth analysis to support their decisions. Their findings can therefore be useful to the USAF in the development of our own Enterprise Ground System. Specifically, EGS-CC has highlighted the importance of a layered architecture using a hierarchical component framework to build composite components that will then be exposed to the system through service oriented architecture (SOA). The SOA would involve an enterprise service bus, much like NASA's GMSEC bus.

2.3.2.4 National Institute for Space Research

Brazil's National Institute for Space Research (INPE) has been running its Satellite Tracking and Control Center (CRC) since the 1980's. In 2008 they began implementing an advanced modernization effort. Today, they offer the only known integrated simulator supporting automated operations [20]. This simulator may not be as robust as other development platform simulators; however, it is capable of simulating many mission-related parameters for all satellites being flown by the ground station. A simulation platform that is extensible to new satellites will be discussed as an essential recommendation going forward.

2.4 Summary

Each case studied in this section presents unique insights into the state-of-the-art as well as emerging approaches to the satellite domain. The non-space domains considered here are excellent targets for this research due either to their maturity or extensive research and development. In addition, the non-space domains were chosen

because they shared many similar goals and requirements of the satellite domain. The space domain examples have much in common with the USAF's goals; they are all government organizations with diverse ranges of satellites. Chapter III discusses how each example was analyzed and what can be learned from them.

III. Methodology

3.1 Chapter Overview

Case study analysis is an established research methodology that examines common trends between multiple projects to determine underlying principles and best practices [21]. Case study analysis is widely used in the software engineering domain [8]. We performed qualitative case study analysis of the non-space and space domains described in the background section in an exploratory manner, for the purpose of identifying actionable recommendations for future satellite ground systems. We identify three software engineering trends in the non-space and satellite operations domains. 1) Layered Architectures, 2) Abstraction, and 3) Integrated Simulation.

3.2 Case Study Design Checklist

The following checklist was taken directly from Host & Runeson, 2007. This checklist was used to validate the design of the case study. Each answer seeks to identify where the corresponding question was addressed within the body of the thesis, though in some cases this is irrelevant.

3.2.1 What is the object of study?

The object of this study is to identify trends and software engineering principles being used in diverse satellite ground stations as well as mature non-space domains to

enable automation. The trends are identified within the body of this chapter and summarized by Table 1 at the end of the chapter.

3.2.2 Is a clear research question or hypothesis defined up front?

As described in Chapter I the research question is, "how can the USAF make future satellite automation more capable and less expensive?" This research question defines the goal of the case study and is summarily answered in Chapter V. The proposed solution is made up of three recommendations presented in Chapter IV.

3.3.3 Is the theoretical basis – relation to existing literature and other cases – defined?

The existence of this checklist compiled from multiple papers on applying case study analysis to software engineering is strong evidence of the widespread use of this technique. Examples of case study analysis in the software engineering domain include; [22], [23], [24]. Therefore, the novelty of this research is the target domain (SATOPS), rather than the case study methodology or its application in software engineering.

3.3.4 Are the author's intentions with the research made clear?

The intention of this research is to take advantage of previously unidentified trends and tools to further the USAF's automation efforts, as described in Chapter I. Specifically, our goal is to identify technical solutions, rather than organizational, training, cultural, or other solutions to automation problems.

3.3.5 Is the case adequately defined (e.g., size, domain, process)?

There are four ground station cases examined in this case study. Each is described in-depth in Chapter II. Three non-space domains were also studied and will be described in-depth in this chapter under section 3.3. Because this is a qualitative analysis, many attributes that would be crucial to a quantitative analysis are less relevant. However, this thesis makes every effort to define key information about each case including maturity and relevance to the target domain.

3.3.6 Is a cause-effect relation under study?

This research is not concerned with a direct cause-effect relation. Rather, it analyzes trends in the SATOPS domain that have been proven to add value. Further, we show that several non-space domains, through experience and extensive research, have identified tools and techniques to make their automation more capable and efficient. These techniques are then shown to be valid in the SATOPS domain.

3.3.7 Will data be collected from multiple sources? Using multiple methods?

This research collected data through a range of methods and sources; specifically, site visits, interviews, and literature review. Foreign ground systems were studied through published words, including books, conference papers and journal articles. US ground stations were studied through limited publications including conference

presentations, but primarily through interviews and site visits. The non-space domains were studied through literature review, again using a combination of books, conference papers and journal articles.

3.3.8 Is there a rationale behind the selection of roles, artifacts, viewpoints, etc.?

The primary criterion for the selection of space-domain examples was that they be highly-respected within the domain as well as similar in scope to the USAF's goals. This was determined through literature review as well as interviews with experts in the space domain. Viewpoints in the non-space domains were selected due to their numerous citations and publications in their respective domains.

3.3.9 Are the case study settings relevant to validly address the research question?

Each case is presented with arguments for its validity in relation to the USAF's automation efforts. Every SATOPS example is government-controlled (either US or foreign), and focused on integrating a wide variety of satellites. The ground systems were also selected for their work in pursuing more capable automation. The non-space domains were selected both for their extensive use of automation, and their operating requirements. Each has many requirements in common with USAF SATOPS, including; reliability, flexibility, scalability, and efficiency.

3.3.10 Is the integrity of the individuals/organizations taken into account?

Each organization that was studied in-person is widely respected and considered to be of the highest integrity. All other sources were peer-reviewed or produced by official US government organizations. Thus, we believe the integrity of each source has been established with a high degree of certainty. There is always the possibility that an organization may publish a work claiming to offer capabilities they do not actually possess. However, given the amount of public scrutiny the organizations reviewed here receive it is unlikely any such deception would be possible.

3.3 Analysis Subjects

The following three domains were each selected for their extensive work in automation. Additionally, they share many requirements with the SATOPS domain. Therefore, the software engineering techniques they employ to improve the capability and efficiency of their automation is likely to be valid in USAF ground system design.

3.3.1 Industrial Control Systems (ICS)

The ICS domain makes the strongest case for layered architectures. Model-View-Controller (MVC) is a layered architecture that has not been widely discussed in the satellite operations domain but is widely used in ICS. Vyatkin argues that MVC is primarily concerned with “design effort” while contributing to “scalability” and

"dependability" [9]. The Controller and Model are the most important parts of MVC. The controller implements a set of functions that are published as services. The Model contains all of the object's behavior. MVC is designed to abstract as much as possible from the functions that the user should be concerned with. Importantly, because the controller and model are connected in a closed loop, the system can be simulated and tested with the operational code. This feature directly supports the T&E domain and simplifies simulation tools. As a result, MVC is a highly attractive approach to use for TT&C architectures. The TT&C can then be connected to an Autonomous Operations Service as well as the other major components of the ground station using the common service oriented architecture employed by GSFC, EGS, and others.

The literature on MVC in industrial control automation presents MVC primarily as an architecture for modeling and simulation (M&S) of industrial systems [25]. However, the features which make MVC ideal for M&S also make it ideal for overcoming the challenges of a common ground architecture TT&C. MVC is organized to abstract the method by which a function is performed on an object in the system. MVC is intended to separate the domain logic and current state of the system (the model) from input logic (the controller) and user input (UI) logic (the view) [26]. The controller serves as the interface layer, providing the view layer with a set list of functions by which it can interact with the model [27]. The value of the controller has become less important over time. There are now version of MVC that merge the Model and Controller [27]. However, this sacrifices some degree of flexibility. In applying MVC to a satellite TT&C, the model would contain a separate object for each satellite, while the controller

would offer the functions that can be performed on those satellites. The controller is little more than an interface layer, but still serves to clearly define the interactions between the model and view. These functions would largely be common amongst satellite buses. For example, station keeping activities, state of health, anomaly detection, and program loading are almost universal in Low Earth Orbit (LEO). Programs to be loaded could also contain common functionality. The controller treats all of these functions the same, passing the command to the correct satellite model which interprets the common function into a tailored function for that satellite and sends the correct commands to hardware. For each new or unique function a satellite requires, the controller must be modified to include the additional functionality. Once the function is added, any satellite object can be modified to perform that function if applicable. ICS also uses integrated simulation in order to improve efficiency. In [28], real-time data was collected in a cellphone manufacturing plant and fed into an event-driven simulator to better predict production outputs. This allowed factory floor managers to make rapid decisions in order to streamline their operations.

3.3.2 Software Test and Evaluation (T&E)

T&E reinforces the value of the MVC pattern for several reasons. First, MVC encourages loose-coupling. The more loosely-coupled a system is, the easier it is to test [13][14]. Second, MVC promotes abstraction. In cases where the potential inputs to a software component vary widely, test automation is more difficult. In order to overcome these difficulties, test automation designers generally abstract away the many variations

of a similar input [13]. They focus the automation on the high-level functions and allow lower-level software to translate common functions into many possible inputs. NASA engineers have published their findings on the importance of model based testing in order to support greater abstraction of the various inputs as well as the underlying structure of the software under testing (SUT), [11]. This is the same approach we are proposing to abstract the differences in the way satellites implement common functionality. Not only does T&E support MVC and abstraction, but it also supports the use of simulation. In many ways, test automation is the same as integrated simulation. The difference between test automation and integrated simulation is that rather than testing the system software, the simulator is testing the automated or manual SATOPS commands. In both cases, the results of these tests are compared with a model that describes how the system should behave [12]. Most importantly, both are looking for deviations in the behavior of the software or satellite that exceed accepted boundaries. In the SATOPS domain, automation scripts, or "procedures" as OMG's SOLM standard calls them, can be adjusted by set parameters to try and find solutions that would keep a satellite's state of health within accepted limits. In T&E, the automated testing simply highlights the problems that must be fixed in the software and generally does not produce and implement possible solutions. Like many fields, T&E is also beginning to use machine learning and artificial intelligence to produce smarter auto-generated test cases and evaluations [15]. This will be addressed in greater detail in Chapter IV, discussing the benefits of increased abstraction on process mining and machine learning.

3.3.3 Intelligent Transportation Systems (ITS)

The ITS domain supports loosely-coupled, layered architectures. MVC is essentially a stand-in for this more generalized concept. Importantly, MVC and layered architectures are not the same as Service Oriented Architectures (SOA). SOA focuses on interfaces and services. Both, MVC and SOA promote loose coupling and common interfaces. In the ITS domain MVC as an architecture is not highlighted specifically. Rather, the principles that MVC supports are identified as essential to integrating heterogeneous components, like vehicles from various manufacturers [16]. This is the same problem faced by the SATOPS domain. The control system must be designed in such a way that individual vehicles or satellites can be modified and re-integrated without requiring changes to the control layer. Current satellite ground stations offer this to some degree, but only within a limited range. Changes to the satellite should not impact other satellites, but they may require complete rewriting of automation scripts. The proper use of MVC-like architectures would minimize the need for adjustments to the automation layer by abstracting the changes that are required to the satellite and focusing on the functionality that the automation is designed to manage.

3.4 Summary

Table 1 summarizes recommendations ICS, T&E, ITS, and other ground stations support. The information in the table is a generalization of what is state of the art and not intended to represent the full-capabilities of each of these ground stations. See section 2.3.2 for a description of the non-USAF ground stations considered here. It is important

to note that many of the trends in ground stations have previously been identified [2] and are being acted upon by the USAF. The trends identified in the non-space domains reinforce and expand upon those in the space domain.

Table 1: Recommendations Supported by Cases

Domain	**Case**	***Layered Architecture***	***Automation Abstraction***	***Integrated Simulation***
Non-Space	ICS	X	X	X
	T&E	X	X	X
	ITS	X	X	X
Space	NASA-GSFC	X		
	NRL-BPTF	X		
	EGS-CC	X		
	INPE-CRC			X

IV. Analysis and Results

4.1 Chapter Overview

This chapter first presents recommendations followed by results of the case study analysis; namely, the use of MVC architectures, integrated simulation capabilities, and increased abstraction. Following these recommendations, this chapter presents an introduction to process mining and an experiment demonstrating the value of process mining in support of automation. Finally, this chapter identifies how increased abstraction would facilitate process mining, leading to transfer learning, resulting in more flexible automation solutions.

4.2 Potential Application Target

The USAF's Multi-Mission Satellite Operations Center (MMSOC), currently being operated at Kirtland AFB and Schriever AFB has been designated the basis of the USAF's next generation of satellite ground service [29] and is the operating platform for the STPSat-2, STPSat-3, and ORS-1 systems. Their stated mission is to provide an affordable ground system for R&D and demo missions and develop innovative solutions to reduce costs [30]. Therefore, MMSOC is an ideal platform for the recommendations presented in this section. However, the principles recommended are generalizable to any ground station aiming to incorporate some level of autonomy into satellite operations.

4.3 Recommendations Introduction

Figure 4 demonstrates the overlapping and complementary nature of the three recommendations presented below. Each recommendation supports or enables the others while directly addressing one of the challenges to automation presented in Chapter I. The full title of each recommendation is; *Implement Model-View-Controller (MVC) for TT&C*, *Integrate Simulators into Operational System Develop,* and *Introduce Greater Abstraction in TT&C and Automation Systems*.

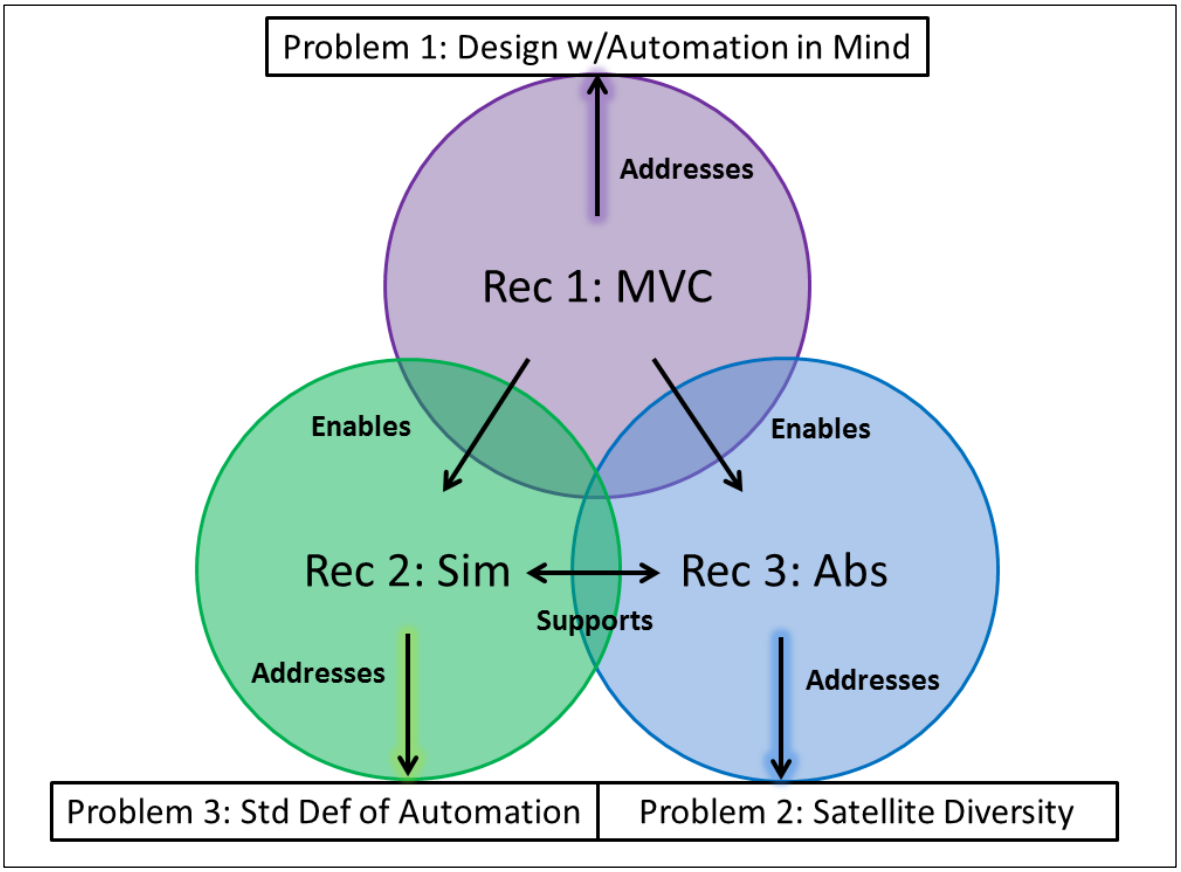


Figure 4. Overview of thesis recommendations

The strongest link between each of the three recommendations proposed by this thesis is the use of models. There are several types of models discussed here. Process models or "pattern-of-life" models will be discussed under Recommendation 3. The value of process models is widely understood and referenced frequently throughout all of the domains studied here, [11], [20], [31]. This is different from the Model layer in the MVC architecture, but they are closely related. The pattern-of-life model describes the high-level activities and inputs to the satellite. The logical model from MVC defines the way those commands are interpreted and communicated to the satellite. A strongly defined model is a crucial element of the MVC design pattern [26]. According to the University of Florida, "Computer simulation is the discipline of designing a model of an actual or theoretical physical system, executing the model on a digital computer, and analyzing the execution output" [32]. Thus, the simulator proposed in Recommendation 2 maintains both "pattern-of-life" models as well as logical models of how subsystems behave when they receive commands. Finally, abstraction is the key element that enables diverse systems to be modeled in the same manner. The Object Management Group, a respected standardization organization, published the Satellite Operations Language Metamodel (SOLM) v1.0 in 2012. This standard defines a universal process modelling language for the space domain. Many organizations, including the USAF Space and Missile Systems Center have taken notice of this document and are working hard to integrate SOLM into their architecture. This is precisely what is recommended by abstraction as presented here. The use of a comprehensive modelling language is crucial to all recommendations presented in this thesis. As such a language exists, any

organization, including the USAF can leverage this to pursue the key recommendations supported here by non-space domains.

4.4 Recommendation 1: Implement Model-View-Controller (MVC) for TT&C

Using a loosely-coupled architecture like MVC is essential for minimizing dependencies [14]. Minimizing dependences simplifies the process of integrating and updating components, speeding up software lifecycles and enabling the system to incrementally incorporate technological advancements, such as the user interface or the underlying artificial intelligence algorithms. Iterative updates address the challenge of legacy subsystems to enable graceful degradation once newer components are available. NASA's GMSEC architecture is an example of applying loose-coupling and layered architecture to the ground station that separates the TT&C from the supporting services. Neptune's Automated Ground Operations (AGO) system is another example of a decoupled design that enabled extensibility to adopt growing autonomous capability.

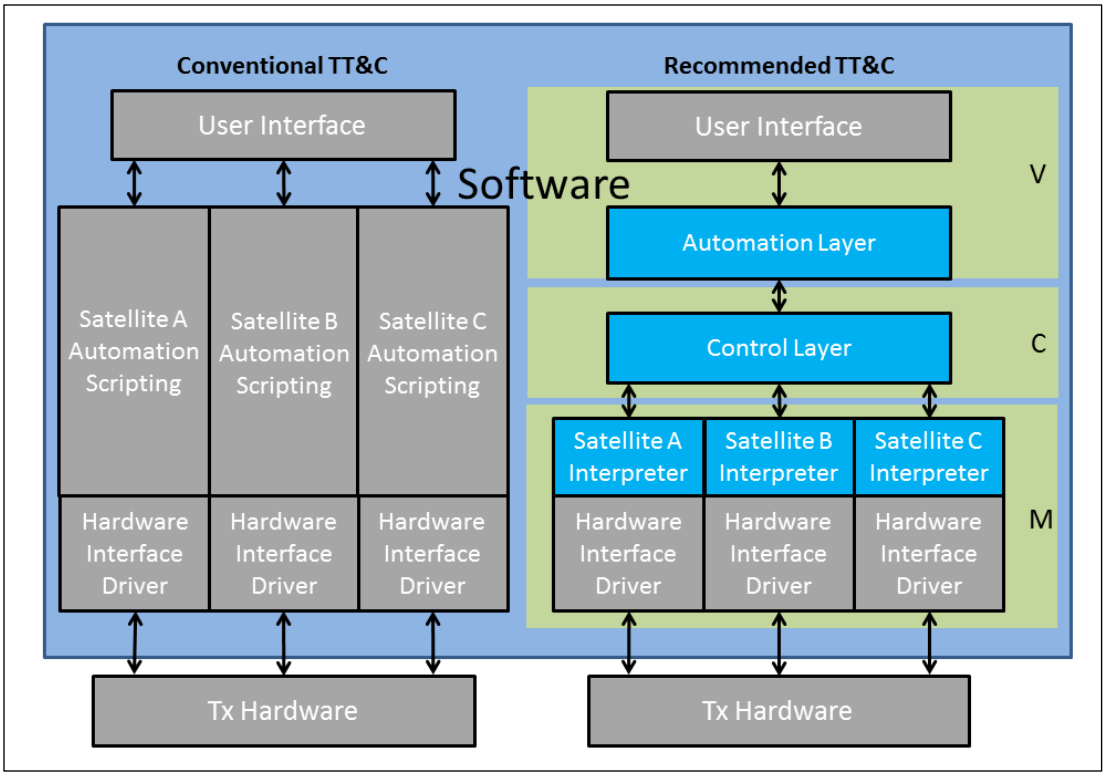


Figure 5. Proposed MVC architecture versus a conventional TT&C

Figure 5 shows a generic, conventional approach to automation scripting, compared to the MVC-like design proposed by this thesis. Components connected without arrows are in the same layer and considered tightly coupled. Components connected with arrows are in different layers and considered loosely coupled. The emphasis in the recommended TT&C is on loose coupling, code reuse and increased abstraction to reduce stove-piping within the architecture. The proposed design is further expanded in Figure 6.

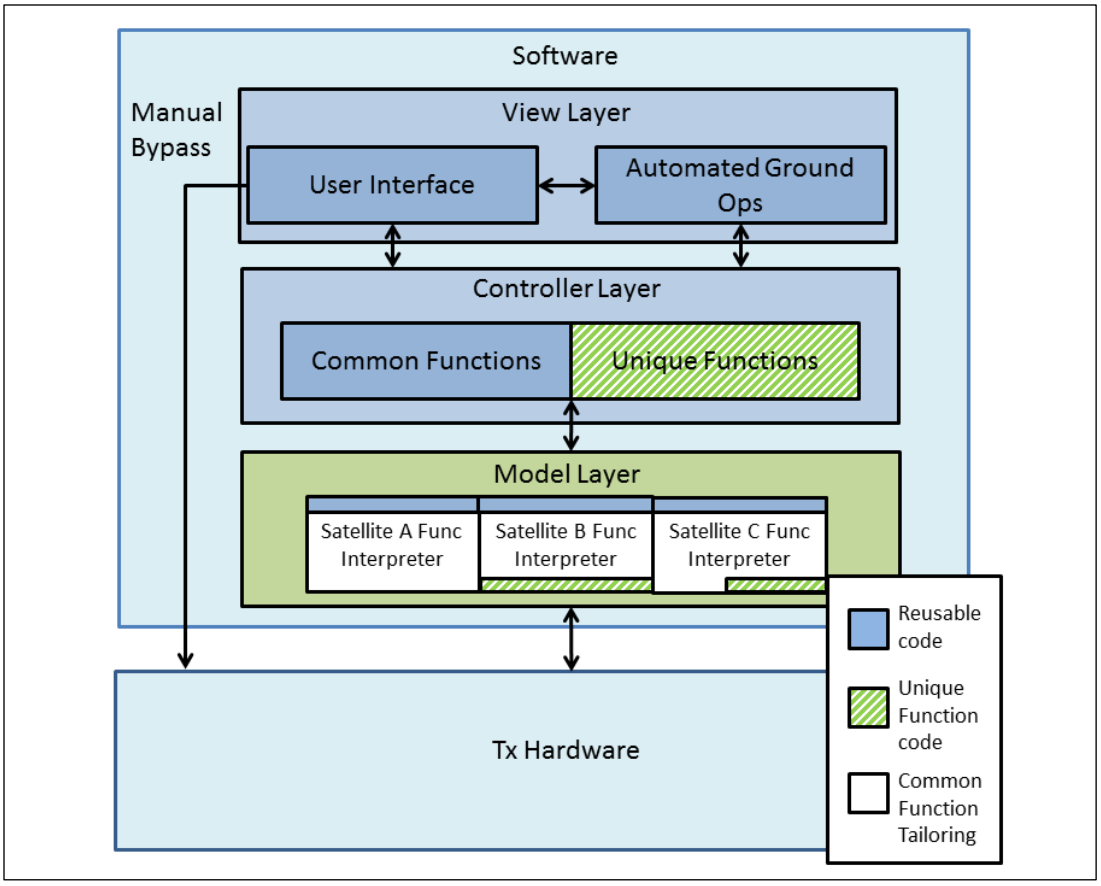


Figure 6. In-depth MVC architecture

Our recommendation—shown in Figure 6—focuses on adopting MVC architecture design principles to enforce loose-coupling on the TT&C. Starting from the layer closest to the satellite operator, the View layer represents both the User Interface (UI) and the automation services that are designed to interact with the TT&C system. Automation services include ground resource management, scheduling, scripted functionality and known anomaly handling. These services will be described in greater detail further on. The Controller is exposed to the rest of the ground station architecture (through a common bus) and provides all of the functions that may be performed on each satellite, whether the command is coming from an automation service or manually

controlled by an operator. These functions are flexible based on input conditions, but provide a buffer that increases abstraction by hiding the vendor specific hardware implementation of these functions for each satellite. The Model contains the driver-like interface between the functionality described in the Controller and the specific low-level instructions for each satellite. As shown in Figure 6, each satellite model contains some reused code that is common across all satellites, most likely associated with the bus. All satellite models should be able to interpret the common functions from the controller. Unique functions in the model layer will require changes to the control layer, though they can be encapsulated to avoid modification of each satellite object when adding new satellite objects.

The MVC architecture enables development of automated services from a specific, abstracted set of instructions, provided by the controller to increase the generality of automation, while the Model actually implements all of the functions. Figure 6 also denotes a manual bypass of the MVC to directly access the low-level hardware in anomaly resolution cases, where the operator needs more direct access to the hardware that may not be common enough to implement in the controller. Figure 6 is focused on the MVC architecture (for TT&C) and does not depict the SOA bus that connects the TT&C to the rest of the ground system. SOA provides a common interface for the TT&C to interact with other services like data management systems, enterprise services, orbital analysis, planning and scheduling, and hardware resource management.

Virtualization, like SOA, is a trend throughout enterprise systems across many domains, including SATOPS. Virtualization is a broad term, encompassing many different design patterns. MVC could be described as a form of virtualization. The

primary difference is that Recommendation 1 is focused on clearly defining and decoupling the TT&C software layers, while virtualization is more about decoupling the TT&C software from TT&C hardware. Because virtualization is already being implemented by the USAF it is not highlighted in these recommendations, despite its close relationship to the MVC design pattern.

4.5 Recommendation 2: Integrate Simulators into Operational System Develop

Simulators are generally standalone components that are developed either after launch or as part of T&E for the satellite. However, incorporating simulators in autonomy development and architectural design addresses the challenge of the expectations and limits for autonomy. The Brazilian ground station INPE integrated their simulator to verify auto-generated mission plans and to reject and update plans that would be unsustainable or harmful based on power systems alone [33].

4.5.1 Simulator development and integration outline

Adding a simulator to the operational ground system could provide a number of benefits. An integrated simulator would lead to greater operator acceptance by offering real-time verification for automated applications. Through a phased implementation, a behavioral simulator could be developed, using real-world data. This phased approach will be discussed in detail further on. As the simulator is refined with real-world data, operators and engineers would be able to see predicted outcomes of their operations and provide input to help refine them. Over some period of time, these predictions would

become increasingly accurate. Organizations should have the flexibility to choose their own thresholds for predictive accuracy. Once the simulator meets these thresholds, operators will have the evidence to trust the simulator's ability to verify automated operations and their outcomes. In the next phase, the simulator would continue to refine its behavioral models and predictions, while verifying autonomous ops plans. When a plan fails this verification, for any number of reasons, it can either be auto-adjusted iteratively until it passes or modified through human intervention.

4.5.2 Addressing lack of standard definition of automation

The proposed simulator would also begin to address the lack of a standard definition of automation by providing a platform against which to test automation and determine its operational limits, providing benchmarks upon which an organization could build a standard understanding of automation and when a human operator needs to intervene. Cases in which automation is not sufficiently capable would become readily apparent. As described above, through the process of prediction and model refinement, some activities and mission parameters would become highly predictable, while others may continue to be much more erratic. Where automation fails to meet mission requirements and predictive thresholds, a clear limit can be identified. Due to the rapidly changing nature of A.I., many of these predictive barriers may only be temporary, but are nonetheless important for producing a comprehensive picture of the current limits of automation.

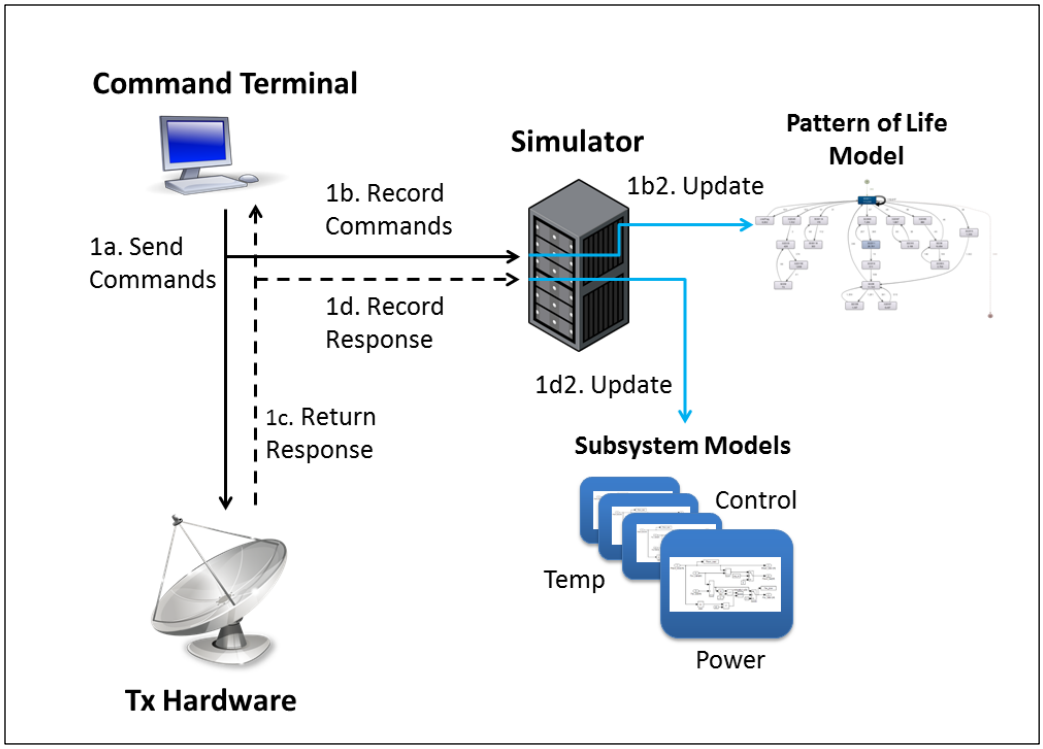


Figure 7. Automated model maintenance through integrated simulation

Having a simulator with accurate, updated models allows applications like automated mission planning to verify plans before use. It would also allow the use of machine learning to develop automatic anomaly responses. Additionally, a concept of operations (CONOPS) planner could be implemented and suggest lifecycle behavior for new satellites based on the body of knowledge gathered from the previous satellites. The CONOPS could also be described as a pattern of life model. Figure 8 demonstrates this process.

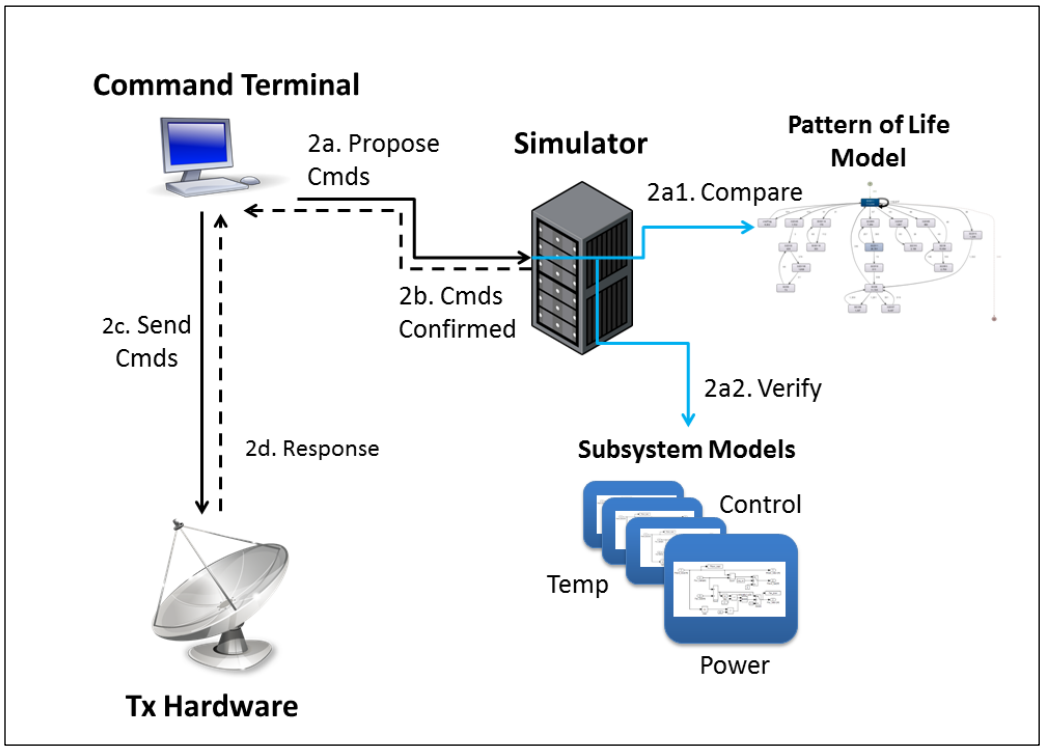


Figure 8. Command confirmation using an integrated simulator

By combining automated CONOPS and mission planning with the proposed MVC architecture, the amount of new code needed to add a satellite to the system would be reduced. In turn this would reduce the cost and time required to add new satellites.

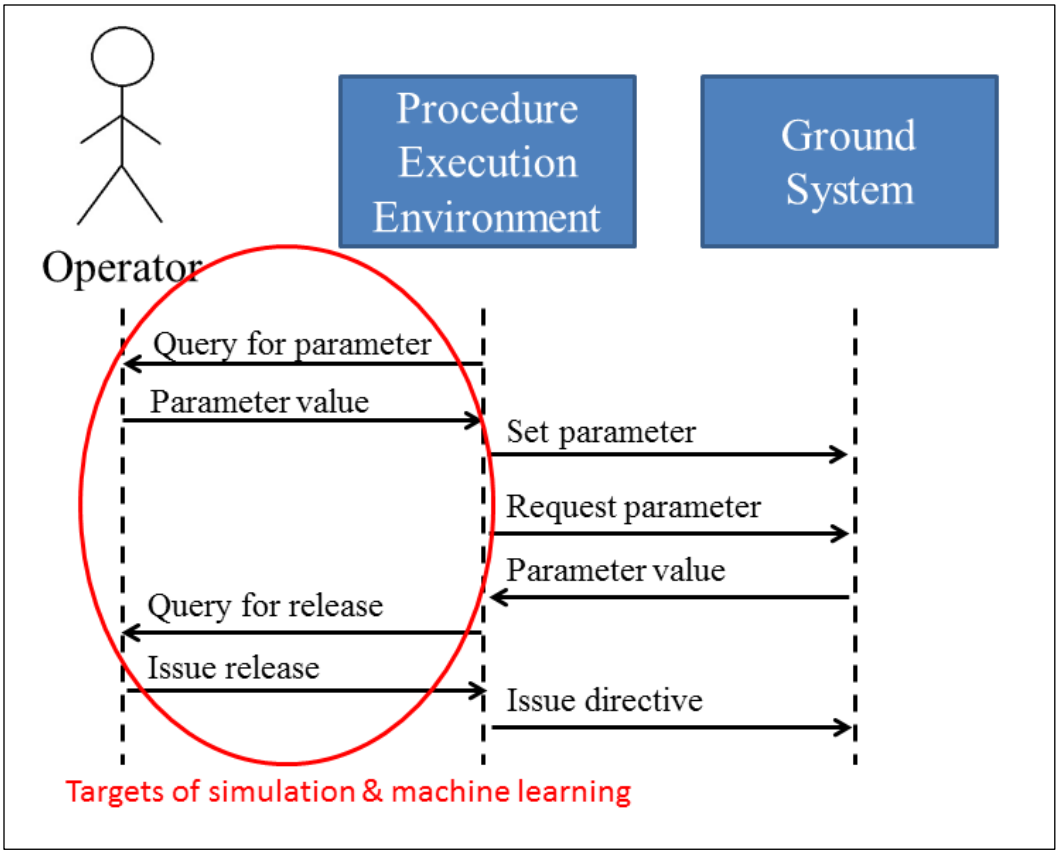


Figure 9. Sequence diagram from SOLM shown with annotations

Figure 9 is a sequence diagram adapted from the SOLM documentation. The red annotations highlight the target of integrated simulation capabilities. The objective of integrated simulation is to develop an extensively trained process model capable of making autonomous decisions, potentially in a manner similar to the NRL's Autonomous Ground Operations system, described in Chapter II. In other words, while the system could be configured to wait for operator approval, the autonomous operations component could make informed decisions all the way from the procedural level to the parameter level. Current space systems do not apply the use of force, in keeping with the Outer Space Treaty [34]. Therefore, such autonomy is still thoroughly in keeping with the

Secretary of Defense's directive, "We will always have a human being in charge of the decision about the use of force" [35]. If future space systems do introduce the use of force, it would be a simple task to limit automation to non-force actions, just as automation can be limited in any number of anomalous circumstances. The capability to minimize operator involvement is directly dependent on the machine learning techniques discussed in the previous paragraph. Those techniques are in turn dependent on the abstraction and process mining techniques discussed in the following section.

4.6 Recommendation 3: Introduce Greater Abstraction in TT&C and Automation Systems

Whereas the MVC recommendation addresses the architectural design, the abstraction recommendation addresses the system diversity challenge. Without introducing greater abstraction, satellites will continue to require completely tailored automation due to the low level differences in every satellite. Increasing abstraction enhances interaction with autonomous systems; first, by identifying the most frequent processes, which are potential candidates to introduce autonomy; second, by minimizing the amount of code that has to be written when a new satellite is added, removed, or changed in the system. This concept is demonstrated in 4.3.1 through the MVC architecture. As shown in Figure 6, properly implemented abstraction and decoupling would greatly reduce the need for changes to the automation layer when changes are made to a satellite. Adding a satellite would require less new code to implement, beyond writing the interface components and function interpreters. Once this is done, the

automation layer can interact with a new satellite in the same way it interacts with every other satellite. Some high-level activities will vary by satellite, but the majority of tightly-coupled scripting would be eliminated.

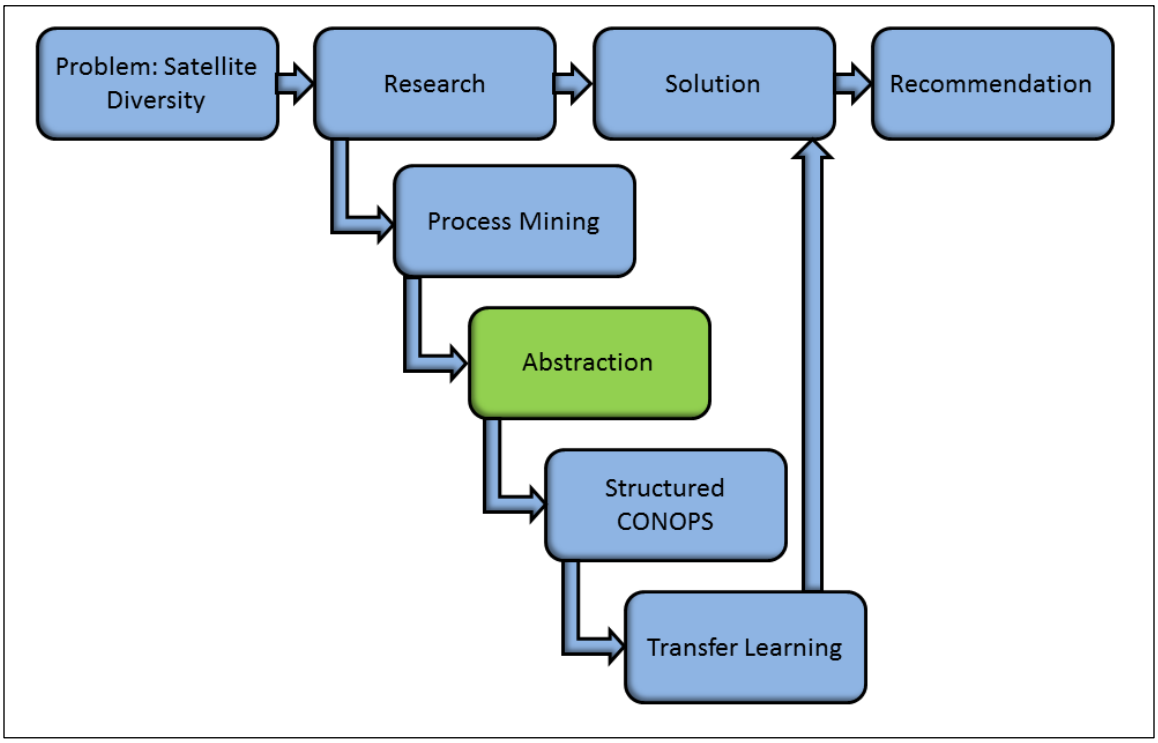


Figure 10. Overview of solution to Satellite Diversity

Figure 9 presents an outline of how abstraction fits into a larger solution to Problem 2 from Chapter I. Satellite diversity is somewhat handled through the proposed MVC architecture, but the next-generation solution is to apply Transfer Learning (TL) to reduce the cost of developing automation for new systems. In order to understand and use TL, we must first be able to perform Process Mining (PM). As will be shown, PM can already be done in the SATOPS domain. However, without implementing greater abstraction, applying PM to current systems will provide limited benefit. Abstraction by itself will not bridge the gap to TL, but it is a prerequisite. Once increased abstraction

has been implemented, mined data will essentially be pre-labelled. The next step beyond this work is to identify the intermediate structure or the order in which activities must occur. This can be reduced to the problem of a structured CONOPS. A structured CONOPS will result in mined processes that can be directly transferred in part or in whole from one satellite to another. The necessity of abstraction to this process will be shown through the following demonstration.

4.6.1 Process Mining Background

PM has been described as "the bridge between data mining and business process modeling" [36]. While data mining largely focuses on characterizing values for objects and system metrics, PM focuses on how the elements of the system work together. PM has been primarily applied to large organizations and businesses to characterize current processes and identify deviations, bottlenecks, and opportunities to increase efficiency. PM offers the space domain a quantitative method of analysis to identify commonality between satellites. Previous papers have highlighted various elements of SATOPS they believed to be common and others that were less so. Such papers relied heavily upon subject matter experts [2]. PM is especially powerful when characterizing complex systems that can be misunderstood by human analysis. Therefore, PM presents an alternative to the previously qualitative analysis of commonality.

The foremost textbook on PM provides one of the most complete explanations of the capabilities of automation. The author describes a continuum of automation made up of three types of processes. On the one end *structured processes* or "Lasagna processes"

are processes in which all activities are "repeatable and have a well-defined input and output [37]. In this type of process, most activities than theoretically be automated. The next category is called *semistructured processes.* These processes have known inputs and their procedures can be outlined, but some human judgement is involved and people can deviate from the model depending on the case. Finally, *unstructured processes* or "Spaghetti processes" the beginning and end states are hard to define. *Unstructured processes* are driven by "experience, intuition, trial-and-error, rules-of-thumb, and vague qualitative information" [37]. These are general guidelines. There is not a hard-and-fast rule for defining spaghetti and lasagna processes. Based on this definition it is clear to see that SATOPS automation can almost never fall into the *unstructured process* category. Every satellite has an established Concept of Operations (CONOPS), even while it is in the testing phase. Thus, SATOPS should never be based on "vague qualitative information". A case could be made to keep SATOPS in the *semistructured* category. This is certainly what much of USAF leadership has in mind. This category also fits the vision of automation laid out by the Secretary of Defense [35]. However, with the rare exception of the use of force, there is little reason SATOPS should not exist almost entirely in the *structured process* region. Human intervention should only be necessary for an anomaly of sufficient magnitude that a capable automation system cannot self-recover.

4.6.2 Impact of Abstraction

Increasing abstraction greatly enhances the value of PM by making the process model portable. Without abstraction, a process model has to be manually labeled. Each command sent to the satellite has to be grouped as part of a larger activity and categorized. Through abstraction, a process model based on one satellite can contain all of the information required to utilize the mined data in building a process model for a new satellite. Abstraction not only increases the usefulness of information contained in the process model, but it reduces the amount of manual intervention required to build a useful model.

4.6.3 Process Mining on SATOPS data

In order to demonstrate the value of PM and increased abstraction, we examined data from several real-world satellites. FalconSAT-3 was launched in 2007 with several experiments and payloads for the USAF Research Laboratory (AFRL). As a small, research satellite, FalconSAT-3 is an excellent, unrestricted target for PM. PropCube 1 and 3, also known as Flora and Merryweather, respectively, were sponsored by the Naval Postgraduate School. They were launched in 2015 and perform ionospheric research. They are newer than FalconSAT-3 and have more working systems, but are quite small and have limited bus capabilities. Even so, they will help demonstrate the commonality or lack thereof in satellites that are very similar. Our hypothesis is that NPS satellites have very similar structure. However, if they are very different, then automation via process mining may be much more sensitive to mission and hardware deviations. If the

latter is the case, then we will see that even activities believed to be very easy to generalize in common ground station architectures may actually be quite difficult. The primary result of such a finding would be that the cost of integrating automation across satellites will be greater and the recommendations presented here will be even more necessary.

The general process followed for each satellite in this experiment involved three steps. First, the command logs were formatted into .txt files. Second, the .txt files were parsed using Python scripts to produce properly organized .csv files that could be imported to the Disco PM tool. Third, the files were imported to Disco and analyzed. The process and scripts used to perform this analysis is shown in greater detail in Appendices A and B.

Disco is a professional PM tool. It is targeted to large businesses with complex business process models, but the developers are quick to point out that it is widely applicable. Disco uses a proprietary process discovery algorithm, though there are many open-source algorithms that would efficiently produce similar results. The only other comparable tool to Disco is called ProM. ProM is largely open-source and designed specifically for in-depth, research analysis. However, it offers significantly more capabilities than are necessary for this experiment. Disco was designed to be simple to use and focuses on intuitive presentation of data and results, making it ideal for a high-level analysis. Disco is capable of interpreting a number of data input formats, though .csv is the simplest. Once a file is loaded, the user simply identifies each column in the data file, including *timestamp*, *row ID*, *event name*, and any other desired values like *resource used*. Disco then analyzes the data to build a process model and generate

multiple statistical reports. The model shows every event along with its frequency count and all of the paths recorded between events along with their frequencies. The process model can then be redrawn to show any percentage of most common events or paths. The model can also be viewed as an average case or a model of the entire data set at once. A more detailed explanation will be presented with the data in the next section. In summary, Disco is a highly flexible tool, with more than enough capability to perform the desired analysis for this research.

4.6.4 FalconSAT-3

We used a Python script to convert the format of FalconSAT-3 command logs into an appropriate format in order to analyze the results with Disco. The first and most obvious application of the process mining results is the ability to identify most common functions. Figure 11 is a visualization created using Disco. In this figure, each box is a command and each arrow represents a transition to another command. The darker blue the box is, the more common the command and the thicker the arrow, the more common the transition. Figure 11 includes manually added labels to identify nondescript command names. Manual labelling would become unnecessary with properly abstracted commands. Figure 11 has been filtered to use only the top third, most frequent commands in order to clearly show the most common sequences. These sequences would provide the highest ROI to implement in a more abstracted manner, [15].

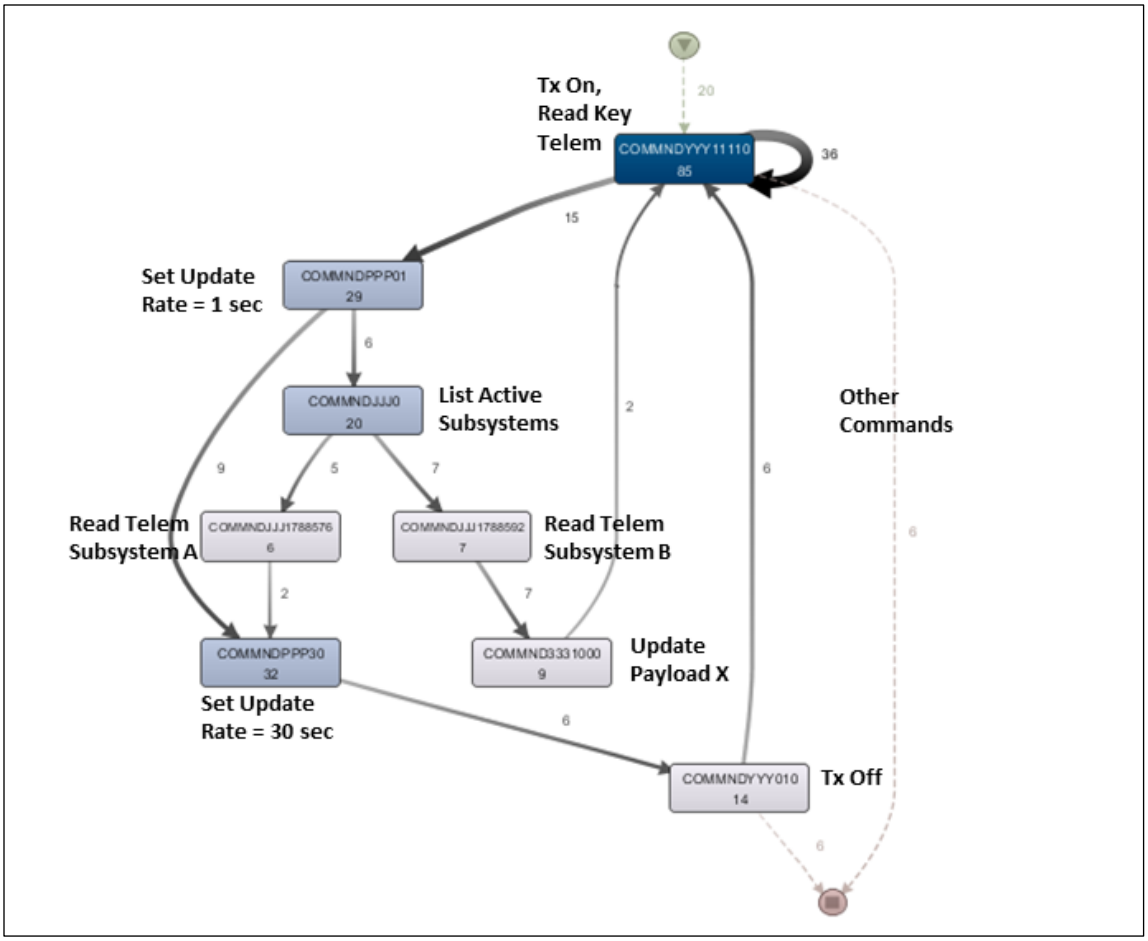


Figure 11. Manually labelled process model of FalconSAT-3

Figure 12 is presented in order to contrast the legibility and simplicity of Figure 11 with an unfiltered, non-annotated model. Figure 12 further demonstrates the value of targeting highest ROI procedures for automation. Note again that darker blue boxes are more common commands and heavier arrows are more common paths.

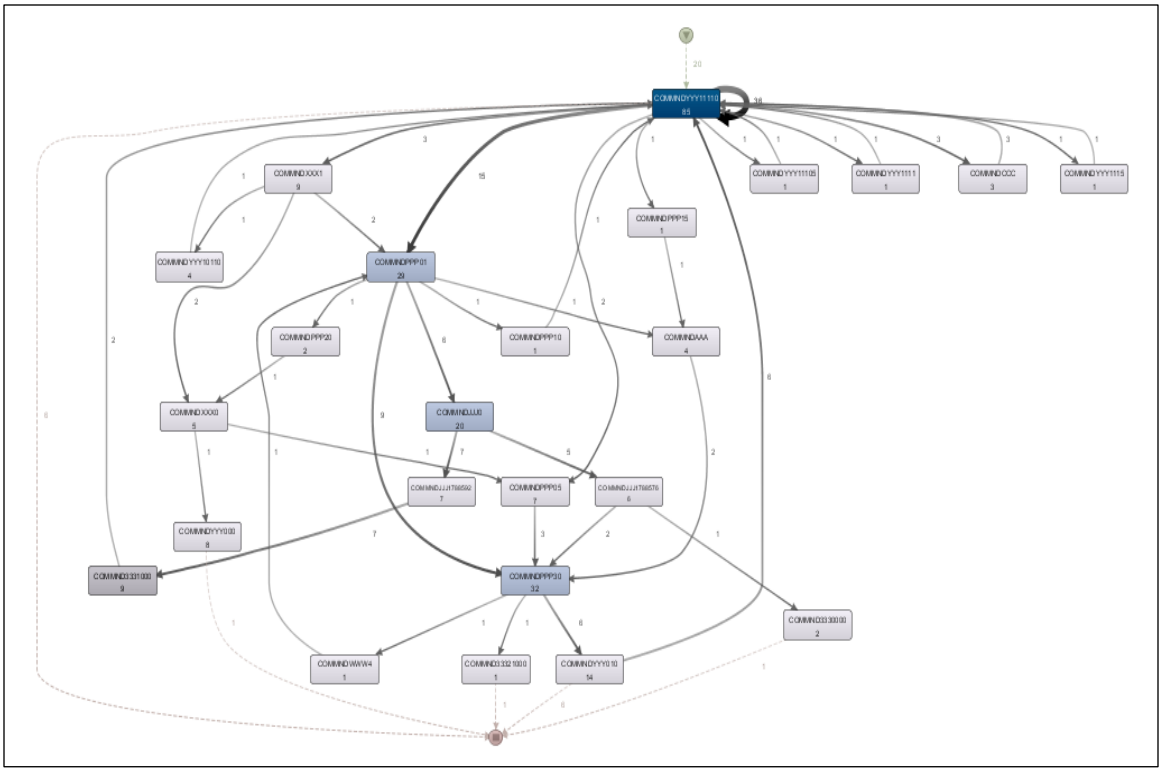


Figure 12. Expanded process model of FalconSAT-3

In this model there are many commands that only occur rarely and often as part of very short sequences. The complexity of this model also serves to demonstrate the relative complexity of even a simple satellite. While being somewhat more capable than an average cube satellite or "cubesat", FalconSAT-3 is already long past its expected lifespan. As a result, most of its experimental payloads are non-functioning. Nevertheless, it still produces a relatively complex process model. Figure 11, which is filtered to only show the top third of commands, ranked by most frequent, looks more like a typical lasagna process. The number of paths is greatly reduced and the relative frequency of each command is quite high. This immediately identifies processes with the highest ROI. However, without the ability to identify what task is performed by the commands, the mined process has very little value to other satellites. By abstracting the

procedures performed by a specific set of commands, the automated SATOPS system can identify generic processes and apply these across satellites. Transfer learning uses many methods to apply data from one domain to another. Each satellite will have a different pattern of life, but each satellite available to the system informs future automation and helps develop general concepts of patterns of life across platforms.

4.6.5 Merryweather, Flora

The second and third examples used in this experiment were the Naval Postgraduate School's cubesats, Merryweather and Flora. These satellites have several significant differences to FalconSAT-3. One of the most significant differences is in the way they record TT&C data. While FalconSAT-3 saves telemetry in a simple .txt file, Merryweather and Flora communicate through network protocols and store data in .pcap files. These proved much more difficult to preprocess. Additionally, these satellites are in contact with ground stations much of the time, despite being in a low Earth orbit. This results in long periods of ground contact with no instruction sequences other than reading telemetry. Because Merryweather and Flora are cubesats, they also do not have propulsion systems to maintain their orbit. This simplifies the recurring commands sent to their buses. The final major difference is that some commands, like reading state of health, are repeated dozens, even hundreds of times in a single contact, without interruption. This repetition can make it more difficult to identify regions of interest manually. For this reason, it was especially important to parse the logs automatically with a Python script that could include all of the data in building a command log for the

Disco software. Figure 13 shows the full process model for Merryweather. Obviously, the model is so large that it is difficult to glean anything from this scale except its relative complexity to FalconSAT-3. We can also see several critical paths that appear to be significant.

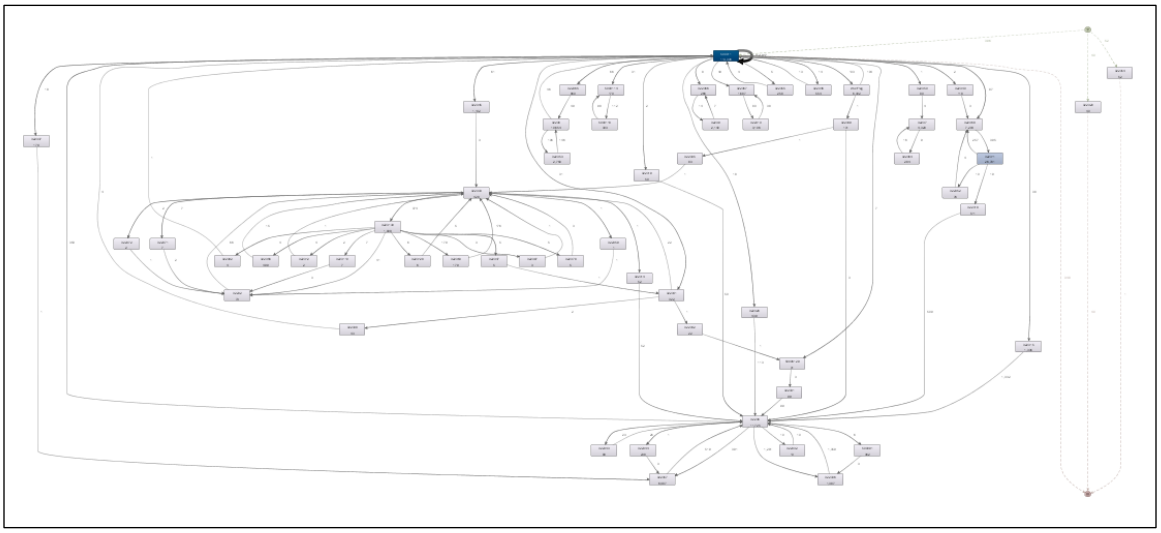


Figure 13. Unreduced Process Model for Merryweather satellite

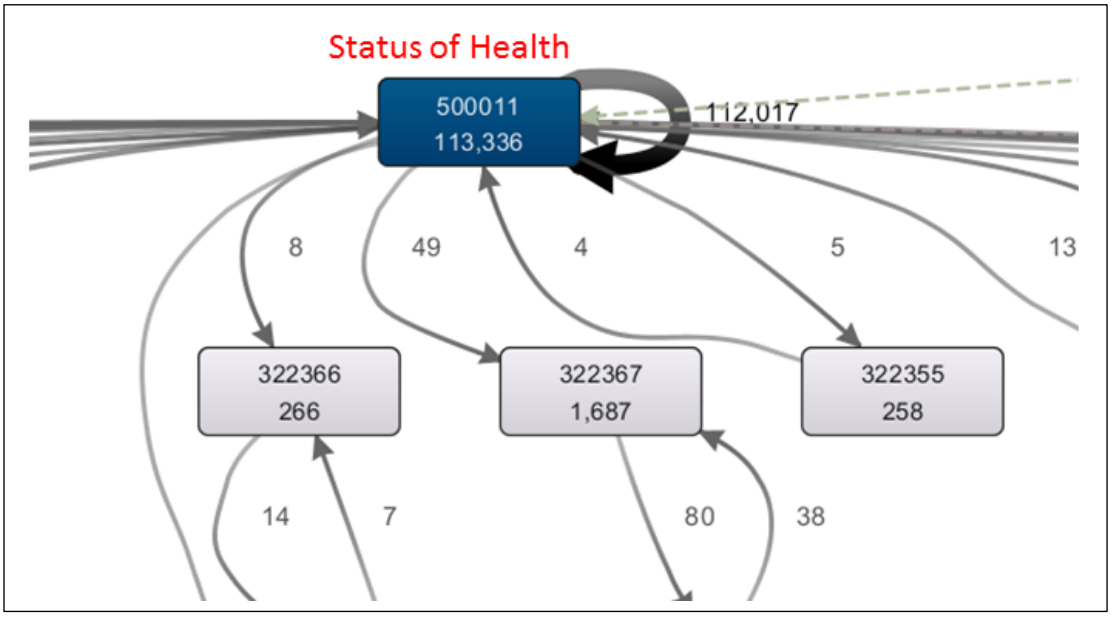


Figure 14. Merryweather state of health command frequency

Figure 14. Merryweather state of health command frequency shows a zoomed-in version of the same model, focused on the most common command. As mentioned above, state of health is overwhelmingly the most common command on Merryweather. It is called 113,000 times in approximately 7,500 hours, or on average, 15 times per hour. The average number of times it is called per contact is 433 according to Disco. This is more than twice the frequency of the second most common command on a case-by-case basis. In total frequency it is called between 4 and 100 times more than any other command. This shows quantitatively the huge ROI delta between automating state of health and all other activities. Such a finding is not surprising. It coincides with many subject matter experts' opinions as published in other papers. More revealing is what process mining can tell us about the other activities and procedures that stand out less dramatically to the naked eye.

Figure 15 is the reduced version of the Merryweather process model. Again, the reduced figure shows the top third most common commands. This model shows 5 distinct groups of commands, each connected to the state of health command. The identified "bad commands" are fragmented commands that were not transmitted or received properly. Another commonality with FalconSAT-3 is that none of these commands are easily identifiable without manual labelling. The higher-level activities being performed by each of the 5 apparent procedures are also not identifiable without additional information. Thus, automation efforts would need to be highly tailored in order to perform them. Abstracting automation and focusing on scripting activities or functions rather than individual commands would greatly increase the commonality of automation.

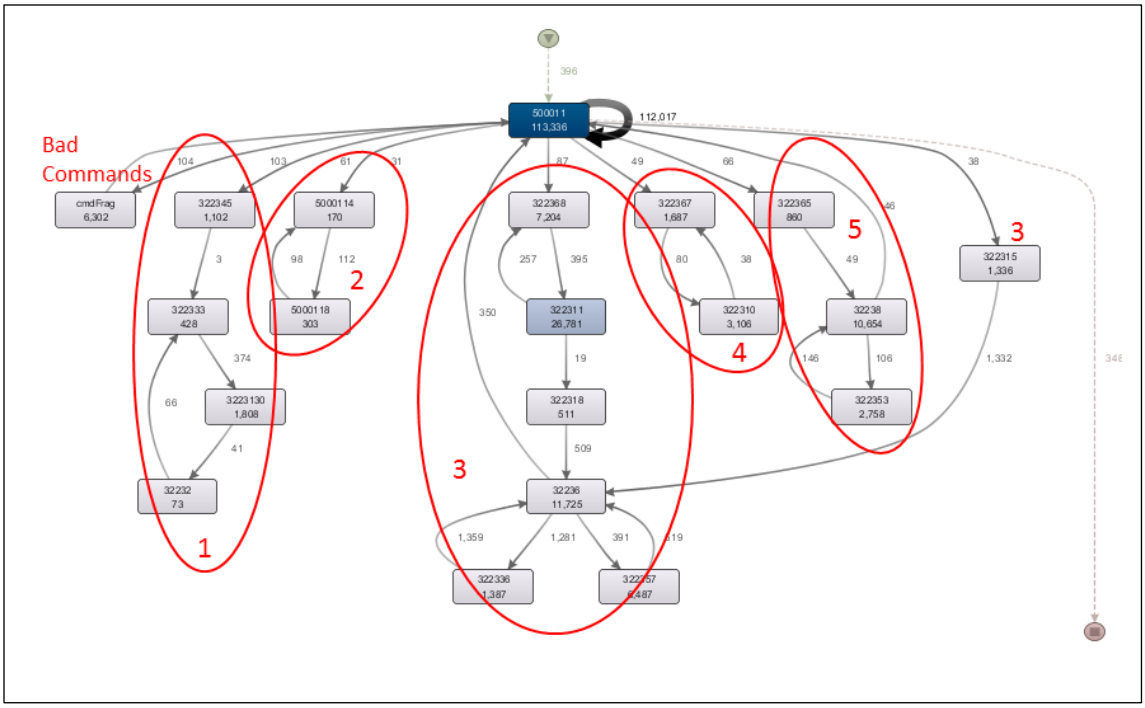


Figure 15. Reduced process model for Merryweather with annotations

Next, we compare the mined process model of Flora with Merryweather. Figure 16 is the unreduced process model for Flora. Again, the scale of the process model does not immediately show anything clearly, other than a different shape. The state-of-health command is still easily identified due to its frequency, but the process appears much more chaotic and less linear. There are still several commands that can be singled out as highly connected or critical nodes in the graph. The most notable metrics are that state of health was called with almost the exact same frequency per case (443) for Flora as it was for Merryweather (433), but the total number of calls was only about 70,000. This is significantly less than Merryweather (113,000) over approximately the same 7,500 hour period, despite the fact that both satellites recorded nearly the same number of passes. Most other commands are also less frequent, suggesting less contact time or less need for

various kinds of updates to and from the satellite. Figure 17 shows the reduced process model.

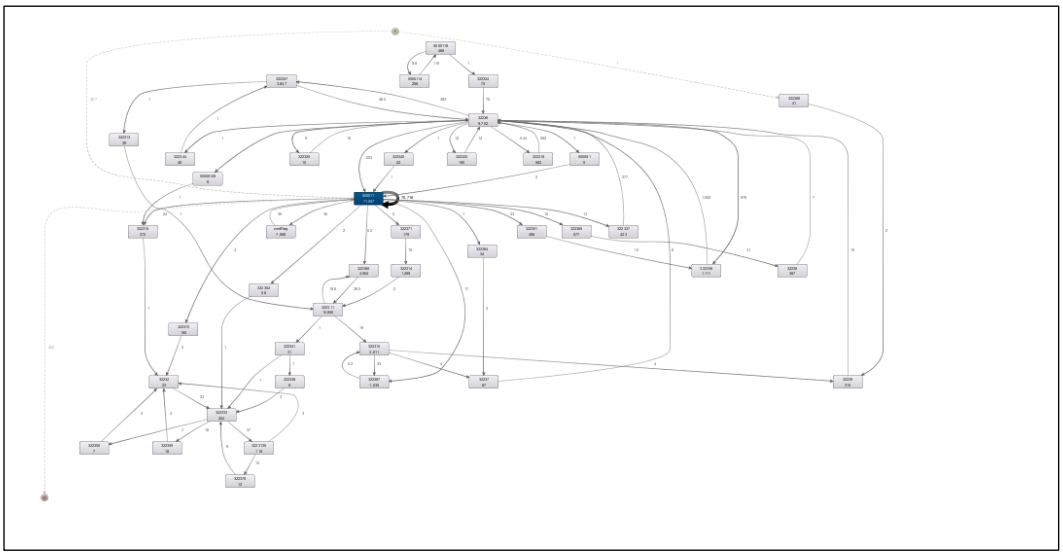


Figure 16. Unreduced process model for Flora satellite

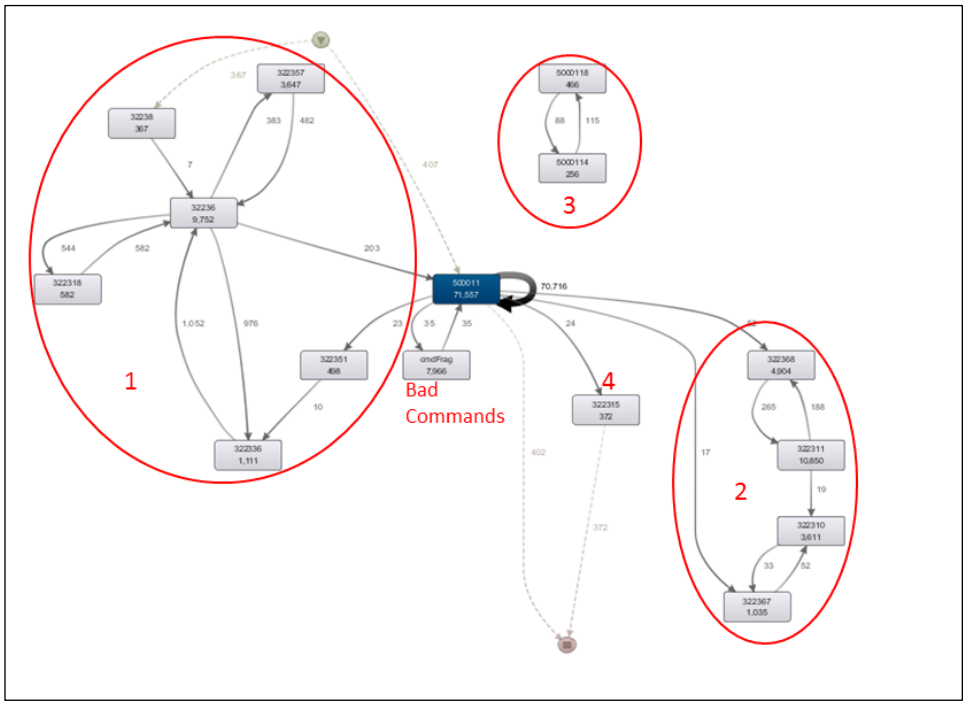


Figure 17. Reduced process model for Flora with annotations

This model has 3 distinct command groups and a fourth solitary command that occurs after the state of health. There were also more "bad" or fragmented commands recorded for Flora (7,900), compared to Merryweather (6,300). This is somewhat surprising given the much larger total number of commands recorded for Merryweather (211,000) versus Flora (125,000).

4.6.6 Process Mining Results Analysis

Flora command group 3 maps directly to Merryweather command group 2, even though they do not appear to connect to the rest of the model in the same way. These are the only commands that the author can identify with total certainty as being part of bus operations, rather than payload. The other groupings, at least at this level, do not match very closely. Thus, without knowing the underlying activities, it would be difficult to leverage pattern of life information from one satellite to the other. This is somewhat surprising given that both satellites are cube satellites, launched at the same time, into the same orbit, with largely identical buses [38]. However, with additional information such as the activities involved, these process models could become transferable in learning patterns of life for both satellites. If these non-matching commands were found to be part of payload operations, it would strongly reinforce the consensus that automation has the highest ROI for bus operations and is largely case-specific for payloads. If all commands were identified as part of bus operations it would illustrate that automation is very difficult to generalize across systems—even nearly identical ones. The current consensus amongst AF Leadership seems to be that integrating "common" bus operations across

satellites is a relatively easy goal. Further research into this field might prove quite the opposite.

PM also revealed that for the limited dataset available on FalconSAT-3, it shared only the state of health/read telemetry activity with the PropCube satellites. None of these example systems are especially complicated, though they do fall into two separate categories of *small sats* and *cube sats*. They still share several characteristics at a superficial level. The primary conclusion of this experiment therefore is that a significant amount of data from a wider variety of satellites would be required to perform the type of transfer learning that will be discussed in the next section. Even so, PM can still provide benefits to the satellite systems being modeled. PM would primarily be supported by Recommendation 2 in delivering more capable automation for each system being modeled through PM.

4.6.7 Transfer Learning in the SATOPS domain

Generic transfer learning happens constantly in almost every domain. The cross-domain case study analysis in this thesis is a simple example. The formal technique of Transfer Learning (TL) is from the Artificial Intelligence (AI) domain. TL is commonly applied to training machine learning algorithms, e.g. neural networks. TL is typically used to improve classification algorithms [39]. There are many methods of performing TL. TL is concerned with deriving information from a source domain and applying it to a target domain. Generally, the source domain is similar in some manner to the target domain. For instance, learning a new language using principles and techniques learned

from a previous language. Another example would be an algorithm that identifies positive book reviews, supplemented with labelled data from movie reviews. Some keywords would likely be used in doth domains to identify good and bad movies or books. Other words might have slightly different meanings. Movies and books match quite closely, but other comparisons would be more difficult. An example cited in literature is that a cell phone review might use the word "small" as a good thing, while a hotel room review would use "small" as a negative word [40]. The most important aspect to TL is that there must be a large amount of labelled data available in the source domain [41], [42]. The goal of TL is to take this large amount of data from the source domain and improve the training of a machine learning algorithm in the target domain. Usually, the method of TL is determined by the circumstances of the source and target domains. One method of TL is known as Hypothesis Transfer Learning (HTL). "HTL is a standard machine learning technique used to train models on separate disjoint training sets, and then transfer the partial models (instead of the data) to reach a unique learning model" [41]. This emphasis on transferring models rather than data is what makes HTL most applicable to the SATOPS domain. As discussed previously, the importance of developing comprehensive models of satellite systems and behaviors within the ground system has been identified in other works [43]. The use of TL to leverage these models across satellites has not been previously proposed. Figure 18 is a generic representation of this process.

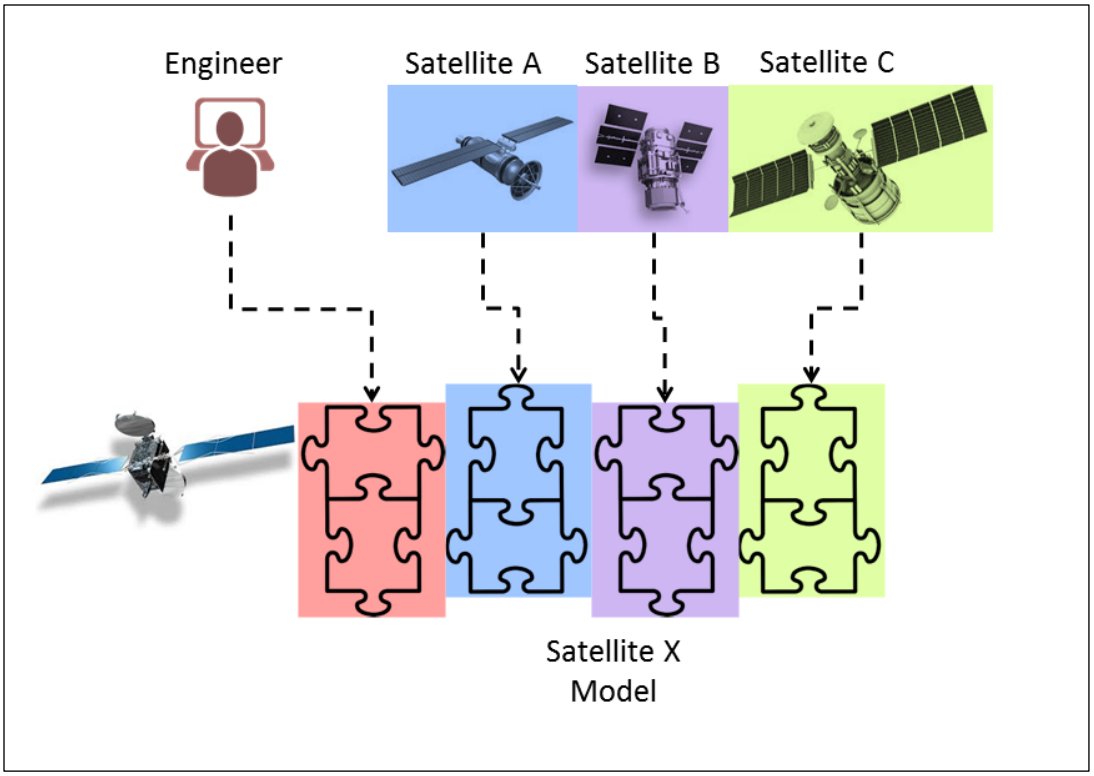


Figure 18. Representation of HTL in SATOPS domain

As mentioned briefly above, one important aspect of TL is the existence of labelled data in the source domain. Section 4.6 showed the necessity for manually labelling existing data in order to create meaningful models. By implementing increased abstraction, this manual labelling would not be necessary. This would produce the labelled data for building partial models that could then be transferred and used to develop models and automated Ops for new satellites. Functionality could be implemented incrementally. Similar to the NRL's AGO, such a service could begin by providing suggestions when a human builds the automated Ops plan for a new satellite. The service would be able to identify potential pattern of life anomalies, recommend common contact outlines, and flag activities that have been left out, but are common to

other satellite Ops plans. Initial targets would include system checks as well as power system and station-keeping functions. As the system matures and operator trust increases, it could be given greater autonomy and provide increasingly useful information through machine learning.

4.7 Generic Ground Station Design

Figure 19 shows a high-level view of how the recommendations in this paper could fit into existing ground station architectures. The component labeled "Telemetry Tracking & Commanding Services" was shown in-depth in Chapter III.

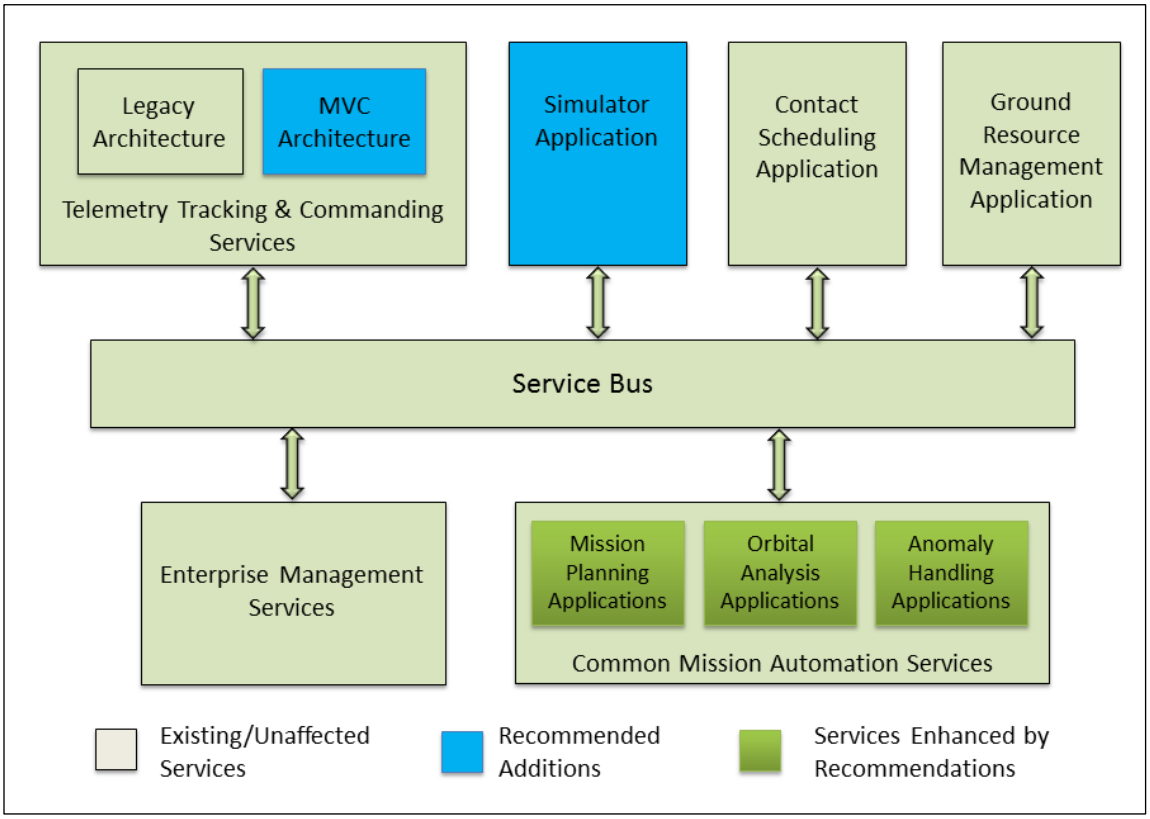


Figure 19. Generic ground station architecture

Notably, SOA is not explicitly described or denoted in the diagram. Several of the ground stations examined as part of the case study already incorporate SOA into the ground scheduling application and ground resource management portion to aid in mission planning. SOA is therefore implied by the common bus. Our contributions are highlighted within the TT&C specifically as well the Simulator service directly connected to the operational system bus. This simulator, discussed in depth in section 4.3.2 could be designed in many different ways and likely in multiple phases of implementation. For the purpose of simplicity it is not deconstructed in this figure.

4.9 Summary

This chapter first assessed each of the three recommendations; layered architecture TT&C, integrated simulators, and increased abstraction. Next it described the background of process mining and how increased abstraction can support process mining. It then demonstrated what is possible with process mining and how this could enable future automation. Finally, a generic architecture with the suggested recommendations was shown.

V. Conclusions and Recommendations

5.1 Chapter Overview

Chapter V presents the conclusions of this research followed by their significance. Next, this chapter presents recommendations for action based on the preceding conclusions. Finally, there are recommendations for future work and a summary.

5.2 Conclusions of Research

The conclusion of this research is that there are software engineering principles in use by non-space domains that can support the USAF's SATOPS automation efforts. Further, these principles, if implemented, have the potential to expand the capabilities of automation beyond current state-of-the-art systems.

5.3 Investigative Questions Answered

- **What are the current trends in satellite operations (SatOps) automation?**
  The current trends in SatOps automation are focused on making ground stations more adaptable and capable of handling diverse satellites. The largest trends are SOA, standardized bus interfaces, virtualization, and software-defined resource pools, and integrated simulators. All of these trends intertwine around the recommendations presented here. They generally focus on methods of achieving loosely-coupled architectures and implementing greater abstraction.

- **Is the USAF keeping pace with these trends?** Many of these trends are being implemented in the USAF's MMSOC ground system. The USAF is working hard to catch up with the more agile develop processes in other organizations. This is understandable due to the nature of USAF acquisitions processes. Additionally, the USAF operates many systems that are much older than MMSOC and have limited funding to pursue modernization.
- **What are the automation trends in mature, non-space domains that might be applicable?** The automation trends in the non-space domains considered here are layered architectures, SOA, standardized interfaces, increasing abstraction, machine learning, and integrated simulation. These trends overlap to a large degree with the SATOPS domain, but their solutions are more varied.
- **How do we know if a mature domain is applicable?** Mature domains were deemed applicable if they have similar requirements, i.e. reliability, safety, flexibility, scalability etc., and a similar pursuit of automation. ICS for example, has very high standards for reliability, scalability, and safety. ICS has also benefitted dramatically from its increased use of automation. T&E has been forced to rely on automation to handle the sheer amount of software being developed. Because software can be intended for almost any domain, all of the software attributes above are necessary. T&E must be as reliable as possible in order to verify the SUT. ITS is probably the domain most concerned with the safety of automation. Millions of humans will be directly dependent on intelligent systems for their transportation in the not-so-distant

future. As a result, automation and AI algorithms will need absolute reliability in order to prevent terrible accidents. As a result, these are all valid domains to leverage in improving automation for the SATOPS domain.

- **How can we make automation less expensive and more flexible?** Layered architectures that enforce loosely-coupled designs, increased abstraction, and machine learning will all support less expensive designs and increase future flexibility. This is the same general progression that has been occurring in software engineering for decades. Higher-level languages and increasing layers of abstraction have made programming increasingly simple and more powerful. These trends will continue for the foreseeable future and the SATOPS domain must continue to follow them. The result will continue to be less expensive, more flexible automation.
- **How do we determine ROI for automation?** ROI for automation is largely determined by frequency of use. The more often an automation feature is required the better the ROI. ROI is also determined by how much automation is required in order to meaningfully decrease the need for human intervention. If through automation an operator only has to press a button one a minute instead of every 10 seconds, this is an improvement. However, as long as the human operator is still required full-time, automation has not produced a significant monetary ROI. Automation that results in lights-out operations has produced a significant ROI. If automation is expanded to include robust anomaly-handling, this will produce another benchmark ROI as engineers are required less-frequently to handle errors. If the principles of abstraction and

integrated simulation recommended here are successfully implemented, it would produce a third benchmark ROI by greatly reducing the amount of time required to integrate new satellite systems and develop their automation.

- **What gives us the highest ROI to automate?** State of health and station-keeping are the two most common operations observed in the limited data sets used for this research. This assessment is reinforced by interviews with subject matter experts. Additionally, each satellite uses several other commands that are quite frequent. These commands may be unique to each satellite, but they offer a high ROI due to their frequency of use. These commands usually involve transmitter settings, telemetry update frequency, GPS signal acquisition, or other simple bus operations. As discussed above, other types of automation, like anomaly handling would also produce significant ROI, but they are not demonstrated by the PM experiment shown in this research.
- **What are the limits of automation?** The limits of automation have not been found. Most limits are self-imposed by organizations. Technology is constantly expanding what is possible with automation. The primary drag on pursuing those limits is cultural and organizational. At the same time, proper design techniques from the ground up are crucial in enabling future tools and techniques to expand automation capabilities. Technological limits to automation do exist and could be found through implementing the recommendations presented here. Even technological limits are changing thanks to massive worldwide investment and research in AI.

5.4 Significance of Research

This research has made three distinct recommendations. MVC and abstraction both have the ability to simplify the process of automation and save the USAF money. Integrated simulation is less about saving money, although there is the potential to save money by creating automatically updated models of satellites. Integrated simulation is primarily focused on building trust, and expanding the capabilities of automation through real-time SATOPS verification. This case study has revealed that the USAF is already leveraging many of the trends in satellite automation, such as SOA, virtualization, and software-defined resource pools. However, there are emergent approaches in other automation domains that have only begun to be recognized in the space sector. As discussed in Chapter II, Brazil's INPE is the only known, operational ground station to have integrated simulation capabilities. This, despite the fact that other automation domains have leveraged integrated simulation with great success.

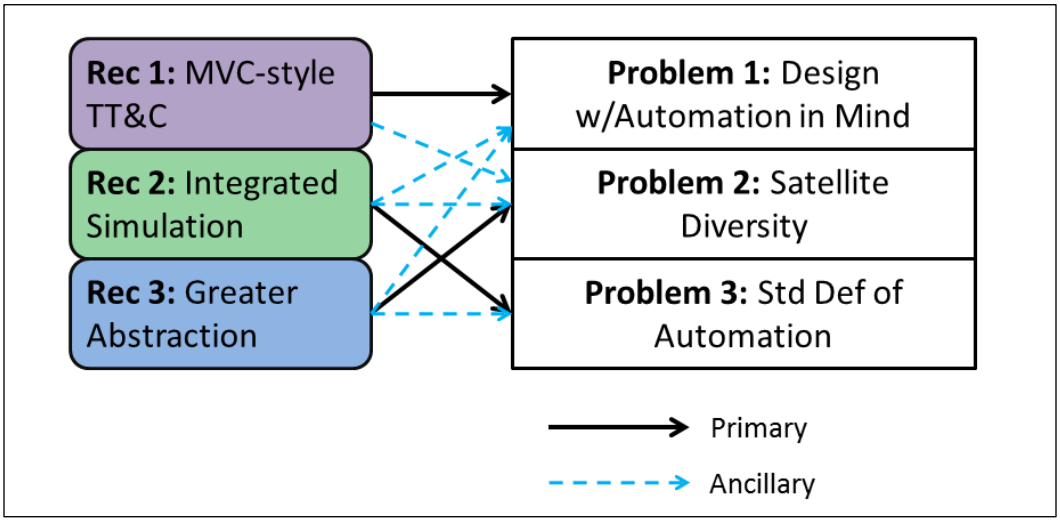


Figure 20. Recommendation-to-problem mapping

Figure 20 illustrates the direct mapping of each recommendation to the problem statement from Chapter I. As shown previously, the recommendations are overlapping and therefore impact each problem to a certain degree. For example, all three recommendations are part of the effort to design a ground system with automation in mind. However, Recommendation 1 is the most implementation specific recommendation and thus primarily addresses Problem 1. Recommendation 2 most closely applies to Problem 3. Developing an integrated simulator would establish the limits of automation through concrete evidence. The simulator would demonstrate which activities can be safely and reliably performed by automation, while identifying activities that may be beyond the capability of current automation. Such activities could become targets of automation in the future as other technologies develop. Recommendation 3 maps very clearly to Problem 2. Abstraction is an almost universal approach to handling diverse implementation problems, as highlighted by each non-space domain in this research.

These results are significant by themselves, but this research is also significant in its novelty. While many organizations—such as the GAO—have performed state-of-the-art surveys of government satellite ground stations in recommending solutions to the U.S. USAF, the Author has found no other literature applying case study analysis from non-space domains to the design of ground station architecture. There is a significant amount of literature cross-applying mathematics, optics, and transmission technologies to the space domain. To the Author's knowledge, there are no other papers examining software engineering principles from non-space domains and applying them to satellite ground

system design, making this research wholly novel from a software engineering perspective.

5.5 Recommendations for Action

By taking advantage of these recommendations, the USAF has the opportunity to save money and expand its automation capabilities in a safe and cost-effective way. Each recommendation will require significant research and development. Applying a variation of the MVC architecture may be the simplest recommendation to implement and leverage. Abstraction as a concept should be applied wherever possible and indeed has made its way into aspects of many ground station architectures, though not in the precise way recommended in this thesis. Integrating simulation capabilities would be a significant and challenging task. Published papers on the subject provide insight into the design of such systems, though actual site visits and hands-on evaluation may be necessary to study existing implementations. Another hurdle for integrated simulation is that some satellite subsystems are more complex to simulate than others. Power system simulation may be one of the easiest. Researchers at the USAF Institute of Technology have found attitude control systems are significantly more difficult than power systems to simulate. The relative difficulties of each element of simulation should inform the process of building robust simulation capabilities. Naturally, the USAF will always be concerned with highest ROI. This may limit what integrated simulation can feasibly provide to ground system architectures until further research and development has advanced the space simulation field.

5.6 Recommendations for Future Research

Due to time and resource constraints, this research has focused on a small set of domains that have each led to important recommendations for ground station automation. As this research appears to be almost entirely novel in academic literature, a logical next step would be to examine additional domains. In addition to adding new domains, continued review of advances in rapidly developing domains, such as vehicle automation, would continue to yield insights. Another direction of future research would be to continue an in-depth examination of each recommendation presented in this thesis. Chapter IV discusses only one recommendation at increased length. This research has only begun to demonstrate the potential capabilities of process mining. Future research should extend this analysis to experimentation with applying transfer learning to the space domain. The other two recommendations, MVC architecture in the TT&C domain and integrated simulation, would be best pursued by researchers with software and astronautical engineering backgrounds.

5.7 Summary

The USAF must leverage software design patterns when designing future ground station architectures. Working systems are not enough if they are not designed with the future in mind. MVC will provide greater flexibility, reducing coupling between system components, and leveraging abstraction. In this way, automation of similar tasks across multiple platforms can be unified. Finally, by integrating M&S capabilities into the system, we can push the limits of automation and begin to address the lack of a standard definition for our automation goals. Future work should consider how to incorporate

automation into new and legacy systems, such as using process mining and transfer learning to leverage past automation efforts and to make automating future similar systems easier. Another area of automation that has a great deal of potential is incident response automation. This type of automation could be designed to handle a wide variety of circumstance from satellite anomalies, to space weather alerts, to cyber security breaches and natural disasters affecting ground systems. Current automation has come far in making systems more reliable and cost-effective. Future automation has the potential to do much more.

Appendix A

The following Python script was used to extract commands from FalconSAT-3 telemetry logs. The output of this script was then input to the process mining tool, Disco. This script was run using Python 3.4.3.

```
import time
startTime = time.clock()
logCounter = 1
outputFile = open('outFile.txt', 'w')

while logCounter <= 20:
    log = open('SatLog' + str(logCounter) + '.txt', 'r', encoding="Latin-1")
    # log.readline()
    logCounter += 1
    bufferStr = ""
    print(log.name)
    for line in log:
        lineList = line.split(" ")
        if lineList[0] == "COMAND0P1S3":
            # print(lineList[3])
            cmd = lineList[3].strip()
            string = bufferStr + ", " + cmd
            outputFile.write(string + "\n")
            outputLogStr = ""
        # print(line, end='')
        bufferList = lineList[0].split("/")
        if len(bufferList) >= 2:
            if len(bufferList[0]) < 2:
               bufferList[0] = "0" + bufferList[0]
            if len(bufferList[1]) < 2:
                bufferList[1] = "0" + bufferList[1]
            lineList[0] = bufferList[1] + "." + bufferList[0] + "." +
            bufferList[2]
        bufferStr = "Pass" + str(logCounter-1) + ", " + lineList[0] + " " +
        lineList[1].strip()
    log.close()

outputFile.close()

endTime = time.clock()
print("Time Elapsed: " + str(endTime-startTime))
```

The following Python script was used to extract relevant data from a comma

separated version of the command logs for the cube satellites, Merryweather and Flora from the Naval Postgraduate School.

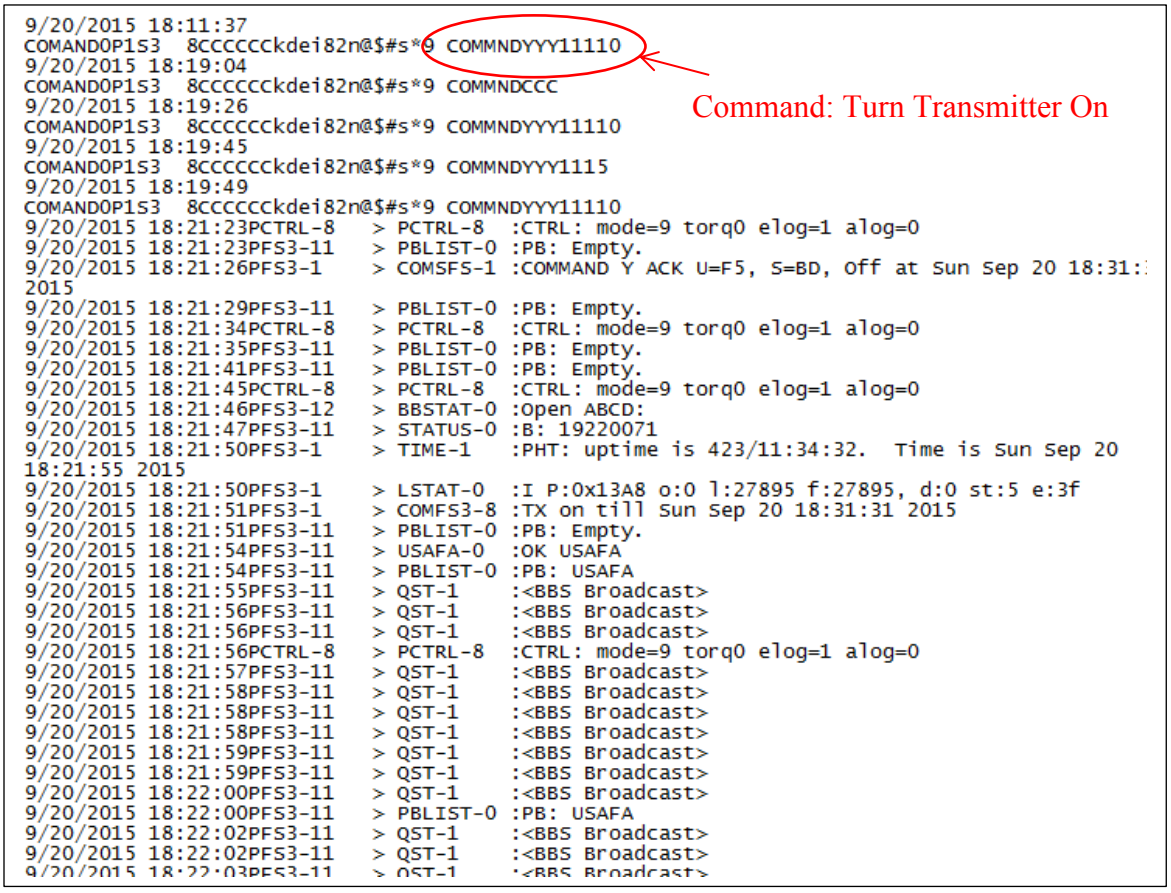
```
import time
import datetime
startTime = time.clock()
initialDTime = datetime.datetime(2016, 1, 1, 0, 0, 0)
fileCounter = 1
passCount = 1
cmdCount = 0
cmdTime = 0

outputFile = open('outFileMerry.txt', 'w')
# outputFile = open('outFileFlora.txt', 'w')


# File breakdown to avoid maximum file sizes.
# while fileCounter <= 2:

# Read in file
# log = open('merryweather' + str(fileCounter) + '.txt', 'r', encoding="Latin-1"
log = open('merryweather' + '.txt', 'r', encoding="Latin-1")
# log = open('flora' + '.txt', 'r', encoding="Latin-1")
# fileCounter += 1
cmd = ""
print(log.name)

# Each line contains a message to/from the satellite
for line in log:
    line = line.replace("\"", "")
    lineList = line.split(",")  # CSV file
    # Ignores header row and only considers messages sent to satellite IP
```

```
30      if lineList[0] != "No." and lineList[3] == [redacted]
31          # print(lineList[3])
32
33          if cmdTime + 1000 < int(float(lineList[1])) and cmdCount > 0:
34              passCount += 1
35              cmdCount = 0
36          cmdTime = int(float(lineList[1]))
37
38          cmdList = lineList[6].split(" ")
39          if lineList[4] == "UDP":
40              cmdList[3] = cmdList[3].replace("Len=", "")
41              cmd = cmdList[2] + cmdList[3]
42          else:
43              cmd = "cmdFrag\n"
44          cmdCount += 1
45          offset = datetime.timedelta(seconds=cmdTime)
46          convDate = initialDTime + offset  # Converted date format
47          string = "Pass" + str(passCount) + ", " + str(convDate) + ", " + cmd
48          outputFile.write(string)
49          # print(line, end='')
50  log.close()
51
52  outputFile.close()
53
54  endTime = time.clock()
55  print("Time Elapsed: " + str(endTime-startTime))
```

IP Address

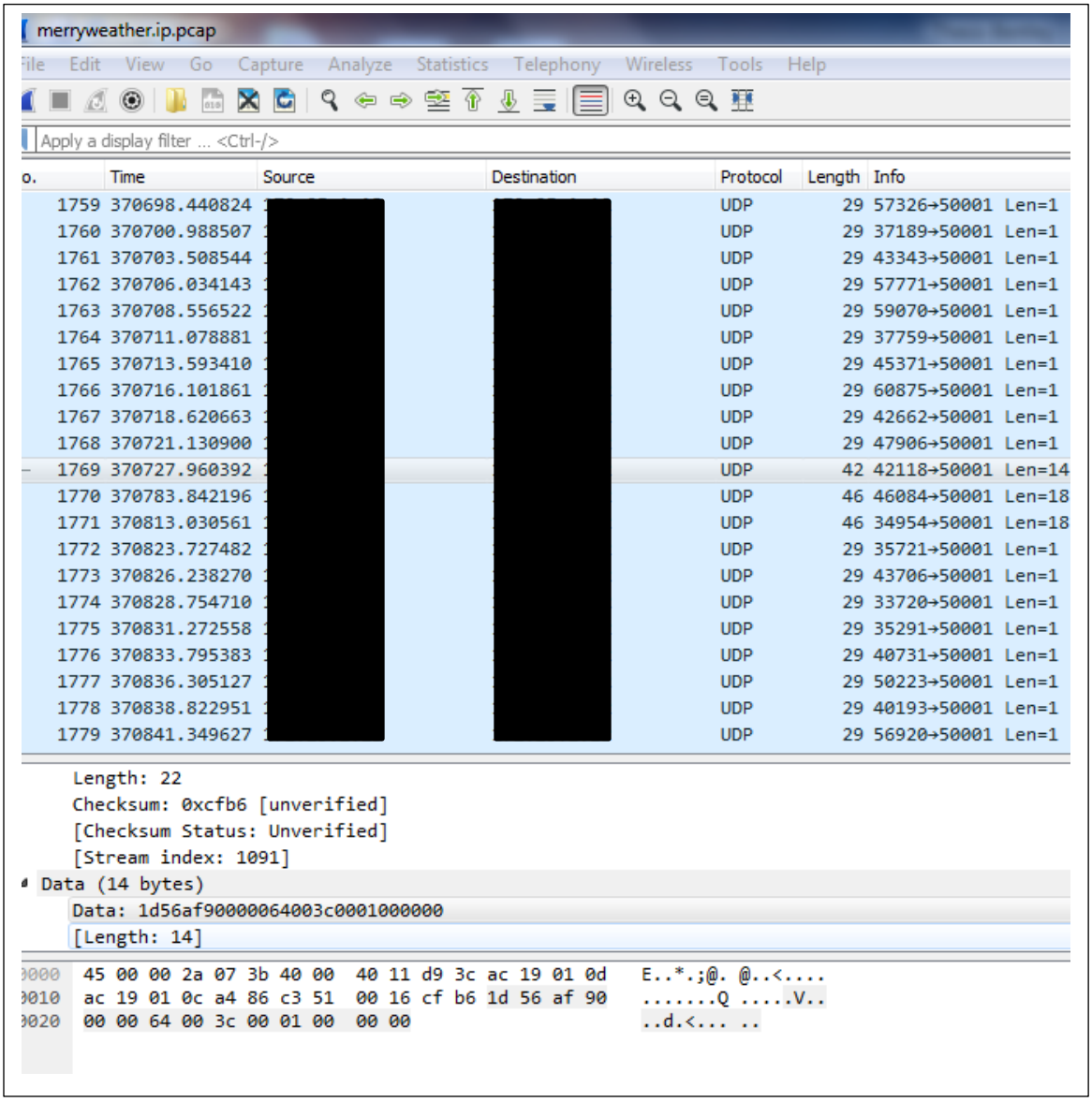
Appendix B

The following are examples of satellite command data. The first is a .txt file from FalconSAT-3.



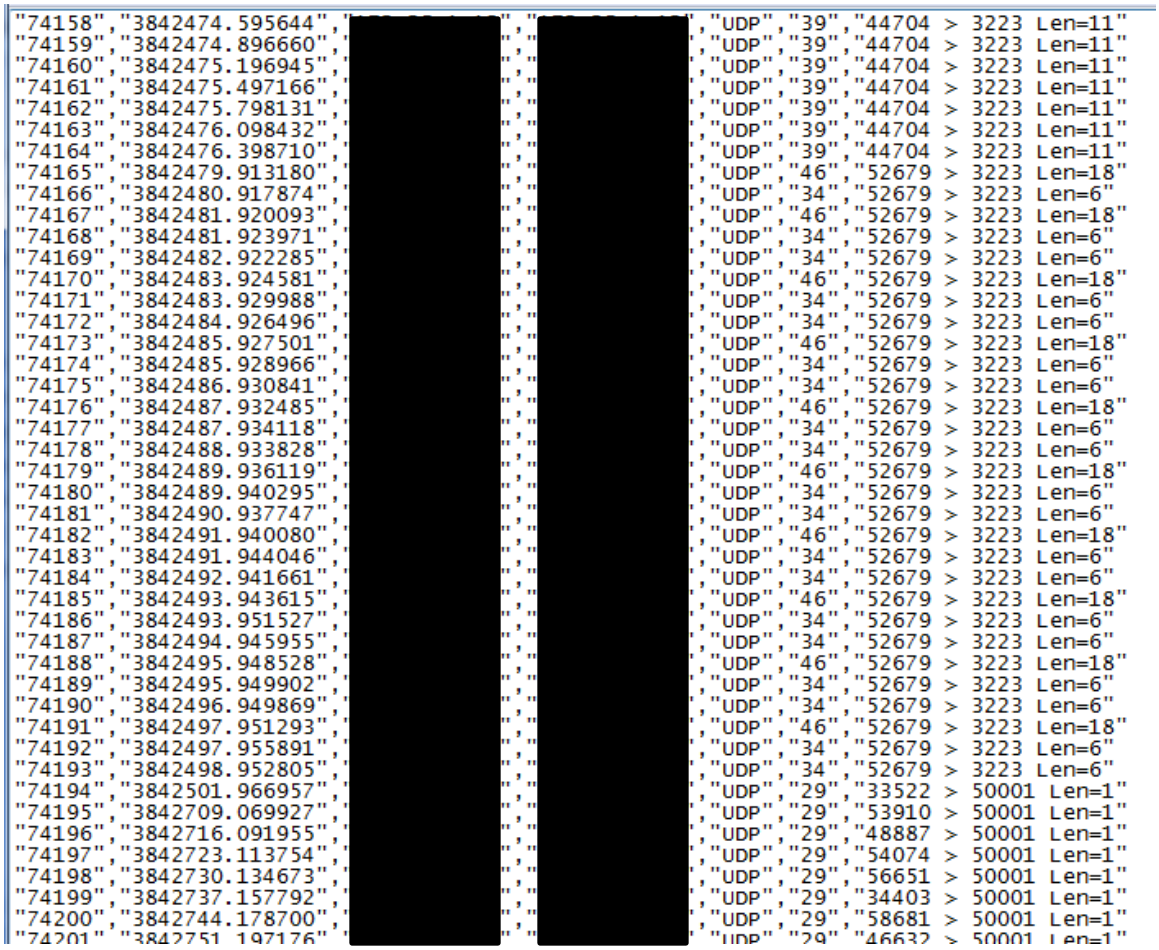

```
9/20/2015 18:11:37
COMANDOP1S3  8CCCCCCkdei82n@$#s*9 COMMNDYYY11110
9/20/2015 18:19:04
COMANDOP1S3  8CCCCCCkdei82n@$#s*9 COMMNDCCC
9/20/2015 18:19:26
COMANDOP1S3  8CCCCCCkdei82n@$#s*9 COMMNDYYY11110
9/20/2015 18:19:45
COMANDOP1S3  8CCCCCCkdei82n@$#s*9 COMMNDYYY1115
9/20/2015 18:19:49
COMANDOP1S3  8CCCCCCkdei82n@$#s*9 COMMNDYYY11110
9/20/2015 18:21:23PCTRL-8    > PCTRL-8  :CTRL: mode=9 torq0 elog=1 alog=0
9/20/2015 18:21:23PFS3-11    > PBLIST-0 :PB: Empty.
9/20/2015 18:21:26PFS3-1     > COMSFS-1 :COMMAND Y ACK U=F5, S=BD, Off at Sun Sep 20 18:31:
2015
9/20/2015 18:21:29PFS3-11    > PBLIST-0 :PB: Empty.
9/20/2015 18:21:34PCTRL-8    > PCTRL-8  :CTRL: mode=9 torq0 elog=1 alog=0
9/20/2015 18:21:35PFS3-11    > PBLIST-0 :PB: Empty.
9/20/2015 18:21:41PFS3-11    > PBLIST-0 :PB: Empty.
9/20/2015 18:21:45PCTRL-8    > PCTRL-8  :CTRL: mode=9 torq0 elog=1 alog=0
9/20/2015 18:21:46PFS3-12    > BBSTAT-0 :Open ABCD:
9/20/2015 18:21:47PFS3-11    > STATUS-0 :B: 19220071
9/20/2015 18:21:50PFS3-1     > TIME-1   :PHT: uptime is 423/11:34:32.  Time is Sun Sep 20
18:21:55 2015
9/20/2015 18:21:50PFS3-1     > LSTAT-0  :I P:0x13A8 o:0 l:27895 f:27895, d:0 st:5 e:3f
9/20/2015 18:21:51PFS3-1     > COMFS3-8 :TX on till Sun Sep 20 18:31:31 2015
9/20/2015 18:21:51PFS3-11    > PBLIST-0 :PB: Empty.
9/20/2015 18:21:54PFS3-11    > USAFA-0  :OK USAFA
9/20/2015 18:21:54PFS3-11    > PBLIST-0 :PB: USAFA
9/20/2015 18:21:55PFS3-11    > QST-1    :<BBS Broadcast>
9/20/2015 18:21:56PFS3-11    > QST-1    :<BBS Broadcast>
9/20/2015 18:21:56PFS3-11    > QST-1    :<BBS Broadcast>
9/20/2015 18:21:56PCTRL-8    > PCTRL-8  :CTRL: mode=9 torq0 elog=1 alog=0
9/20/2015 18:21:57PFS3-11    > QST-1    :<BBS Broadcast>
9/20/2015 18:21:58PFS3-11    > QST-1    :<BBS Broadcast>
9/20/2015 18:21:58PFS3-11    > QST-1    :<BBS Broadcast>
9/20/2015 18:21:58PFS3-11    > QST-1    :<BBS Broadcast>
9/20/2015 18:21:59PFS3-11    > QST-1    :<BBS Broadcast>
9/20/2015 18:21:59PFS3-11    > QST-1    :<BBS Broadcast>
9/20/2015 18:22:00PFS3-11    > QST-1    :<BBS Broadcast>
9/20/2015 18:22:00PFS3-11    > PBLIST-0 :PB: USAFA
9/20/2015 18:22:02PFS3-11    > QST-1    :<BBS Broadcast>
9/20/2015 18:22:02PFS3-11    > QST-1    :<BBS Broadcast>
9/20/2015 18:22:03PFS3-11    > QST-1    :<BBS Broadcast>
```

Other satellite data requires some degree of preprocessing to be viewable. The following is command data from the Naval Postgraduate School's PropCube program. Specifically, the Merriweather cube satellite. This data is in a .pcap file format and is viewed using the network tool, Wireshark.



Wireshark allows the user to export a .pcap file in multiple formats. First, the .pcap was exported as .csv file (which only includes the information shown inline for each row. The .csv file is shown below. This file was then parsed using the Python script shown in

Appendix A. By using a combination of the destination IP, port number, and command length, each command can be uniquely identified without reading the data line.



```
"74158","3842474.595644",[redacted],[redacted],"UDP","39","44704 > 3223 Len=11"
"74159","3842474.896660",[redacted],[redacted],"UDP","39","44704 > 3223 Len=11"
"74160","3842475.196945",[redacted],[redacted],"UDP","39","44704 > 3223 Len=11"
"74161","3842475.497166",[redacted],[redacted],"UDP","39","44704 > 3223 Len=11"
"74162","3842475.798131",[redacted],[redacted],"UDP","39","44704 > 3223 Len=11"
"74163","3842476.098432",[redacted],[redacted],"UDP","39","44704 > 3223 Len=11"
"74164","3842476.398710",[redacted],[redacted],"UDP","39","44704 > 3223 Len=11"
"74165","3842479.913180",[redacted],[redacted],"UDP","46","52679 > 3223 Len=18"
"74166","3842480.917874",[redacted],[redacted],"UDP","34","52679 > 3223 Len=6"
"74167","3842481.920093",[redacted],[redacted],"UDP","46","52679 > 3223 Len=18"
"74168","3842481.923971",[redacted],[redacted],"UDP","34","52679 > 3223 Len=6"
"74169","3842482.922285",[redacted],[redacted],"UDP","34","52679 > 3223 Len=6"
"74170","3842483.924581",[redacted],[redacted],"UDP","46","52679 > 3223 Len=18"
"74171","3842483.929988",[redacted],[redacted],"UDP","34","52679 > 3223 Len=6"
"74172","3842484.926496",[redacted],[redacted],"UDP","34","52679 > 3223 Len=6"
"74173","3842485.927501",[redacted],[redacted],"UDP","46","52679 > 3223 Len=18"
"74174","3842485.928966",[redacted],[redacted],"UDP","34","52679 > 3223 Len=6"
"74175","3842486.930841",[redacted],[redacted],"UDP","34","52679 > 3223 Len=6"
"74176","3842487.932485",[redacted],[redacted],"UDP","46","52679 > 3223 Len=18"
"74177","3842487.934118",[redacted],[redacted],"UDP","34","52679 > 3223 Len=6"
"74178","3842488.933828",[redacted],[redacted],"UDP","34","52679 > 3223 Len=6"
"74179","3842489.936119",[redacted],[redacted],"UDP","46","52679 > 3223 Len=18"
"74180","3842489.940295",[redacted],[redacted],"UDP","34","52679 > 3223 Len=6"
"74181","3842490.937747",[redacted],[redacted],"UDP","34","52679 > 3223 Len=6"
"74182","3842491.940080",[redacted],[redacted],"UDP","46","52679 > 3223 Len=18"
"74183","3842491.944046",[redacted],[redacted],"UDP","34","52679 > 3223 Len=6"
"74184","3842492.941661",[redacted],[redacted],"UDP","34","52679 > 3223 Len=6"
"74185","3842493.943615",[redacted],[redacted],"UDP","46","52679 > 3223 Len=18"
"74186","3842493.951527",[redacted],[redacted],"UDP","34","52679 > 3223 Len=6"
"74187","3842494.945955",[redacted],[redacted],"UDP","34","52679 > 3223 Len=6"
"74188","3842495.948528",[redacted],[redacted],"UDP","46","52679 > 3223 Len=18"
"74189","3842495.949902",[redacted],[redacted],"UDP","34","52679 > 3223 Len=6"
"74190","3842496.949869",[redacted],[redacted],"UDP","34","52679 > 3223 Len=6"
"74191","3842497.951293",[redacted],[redacted],"UDP","46","52679 > 3223 Len=18"
"74192","3842497.955891",[redacted],[redacted],"UDP","34","52679 > 3223 Len=6"
"74193","3842498.952805",[redacted],[redacted],"UDP","34","52679 > 3223 Len=6"
"74194","3842501.966957",[redacted],[redacted],"UDP","29","33522 > 50001 Len=1"
"74195","3842709.069927",[redacted],[redacted],"UDP","29","53910 > 50001 Len=1"
"74196","3842716.091955",[redacted],[redacted],"UDP","29","48887 > 50001 Len=1"
"74197","3842723.113754",[redacted],[redacted],"UDP","29","54074 > 50001 Len=1"
"74198","3842730.134673",[redacted],[redacted],"UDP","29","56651 > 50001 Len=1"
"74199","3842737.157792",[redacted],[redacted],"UDP","29","34403 > 50001 Len=1"
"74200","3842744.178700",[redacted],[redacted],"UDP","29","58681 > 50001 Len=1"
"74201","3842751.197176",[redacted],[redacted],"UDP","29","46632 > 50001 Len=1"
```

Bibliography

[1] Report, “Scientific Advisory Board Report on A Space Roadmap for the 21 st Century Aerospace Force,” 2000.

[2] GAO, “SATELLITE Long-Term Planning and Adoption of Commercial Practices Could Improve DOD ’ s Operations,” 2013.

[3] “Spacecraft: Platforms,” *Gunter’s Space Page*, 2016. [Online]. Available: http://space.skyrocket.de/directories/sat_bus.htm. [Accessed: 04-Oct-2016].

[4] “Planet Labs Specifications : Spacecraft Operations & Ground Systems,” 2015.

[5] U. Mandelbaum, “Satellite Operations Automation,” pp. 1–15, 2016.

[6] “Automation,” *IEEE Robotics and Automatoin Society*, 2016. [Online]. Available: www.ieee.org. [Accessed: 06-Oct-2016].

[7] P. Klein, “Automated Ground Operations,” no. May, 2010.

[8] M. Host and P. Runeson, “Checklists for Software Engineering Case Study Research,” *Proc. - 1st Int. Symp. Empir. Softw. Eng. Meas. ESEM 2007*, pp. 482–484, 2007.

[9] V. . B. Vyatkin, “Software engineering in industrial automation: State-of-the-art review,” *IEEE Trans. Ind. Informatics*, vol. 9, no. 3, pp. 1234–1249, 2013.

[10] B. Vogel-Heuser, C. Diedrich, A. Fay, S. Jeschke, S. Kowalewski, M. Wollschlaeger, and P. Göhner, “Challenges for Software Engineering in Automation,” *J. Softw. Eng. Appl.*, vol. 7, no. May, pp. 440–451, 2014.

[11] D. Ganesan, M. Lindvall, S. Hafsteinsson, R. Cleaveland, S. L. Strege, and W. Moleski, “Experience Report: Model-Based Test Automation of a Concurrent Flight Software Bus,” *2016 IEEE 27th Int. Symp. Softw. Reliab. Eng.*, pp. 445–

454, 2016.

[12] I. Fernandez, A. Di Cerbo, E. Dehnhardt, and M. Tipaldi, “Test automation for critical space software,” *3rd IEEE Int. Work. Metrol. Aerospace, Metroaerosp. 2016 - Proc.*, pp. 551–555, 2016.

[13] D. Graham and M. Fewster, *Experiences of Test Automation*. 2012.

[14] M. Feathers, *Working Effectively With Legacy Code*. 2005.

[15] Y. Amannejad, V. Garousi, R. Irving, and Z. Sahaf, “A search-based approach for cost-effective software test automation decision support and an industrial case study,” *Proc. - IEEE 7th Int. Conf. Softw. Testing, Verif. Valid. Work. ICSTW 2014*, pp. 302–311, 2014.

[16] Z. A. Polgar and A. C. Hosu, “Load Balancing Mechanism in Heterogeneous Networks for Intelligent Transportation Systems Based on Network Level Context Information,” *39th Int. Conf. Telecommun. Signal Process.*, pp. 20–26, 2016.

[17] J. Cleveland and B. Cassidy, “Neptune Software Introduction,” 2015.

[18] M. Molina, “Three Integration Architectures for Satellite and Ground Operations Automation,” *SpaceOps 2012 Conf.*, 2012.

[19] C. Cruzen, M. Schmidhuber, and L. Dubon, *Space Operations: Innovations, Inventions, and Discoveries*. 2015.

[20] J. Tominaga, J. Silva, and M. Ferreira, “A Proposal for Implementing Automation in Satellite Control Planning,” *SpaceOps 2008 Conf.*, pp. 1–9, 2008.

[21] K. M. Eisenhardt and M. E. Graebner, “Building theories from case study research,” *Acad. Manag. Rev.*, vol. 14, no. 4, pp. 532–550, 1989.

[22] A. Taweel, B. Delaney, T. N. Arvanitis, and L. Zhao, “Communication,

knowledge and co-ordination management in globally distributed software development: Informed by a scientific software engineering case study," *Proc. - 2009 4th IEEE Int. Conf. Glob. Softw. Eng. ICGSE 2009*, pp. 370–375, 2009.

[23] M. J. Khan, "Applications of case-based reasoning in Software Engineering: a systematic mapping study," *IET Softw.*, vol. 8, no. 6, pp. 258–268, 2014.

[24] K. Ravindran and S. Ramesh, "Model-based engineering of cyber-physical software systems for smart worlds: A case study of automobile control systems," *Proc. 2013 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI 2013*, pp. 1710–1717, 2013.

[25] J. H. Christensen, "Design patterns for systems engineering with IEC 61499," 2000.

[26] Microsoft, "ASP.NET MVC Overview," *Microsoft Developer Network*, 2014. [Online]. Available: http://msdn.microsoft.com/en-us/library/dd381412(v=vs.108).aspx. [Accessed: 21-Dec-2016].

[27] S. Borini, *Understanding Model-View-Controller*. 2015.

[28] M. Kitazawa, S. Takahashi, T. B. Takahashi, A. Yoshikawa, and T. Terano, "Improving a Cellular Manufacturing System through Real Time-Simulation and-Measurement," *2016 IEEE 40th Annu. Comput. Softw. Appl. Conf.*, pp. 117–122, 2016.

[29] M. Gruss, "U.S. Moving Toward Common Satellite Operating Architecture," *Space News*, 2015. [Online]. Available: http://spacenews.com/u-s-moving-toward-common-satellite-operating-architecture/. [Accessed: 11-Oct-2016].

[30] K. M. Kitani, Y. Sato, and A. Sugimoto, "Recovering the basic structure of human

activities from a video-based symbol string," *2007 IEEE Work. Motion Video Comput. WMVC 2007*, 2007.

[31] C. Yang, V. Vyatkin, and C. Pang, "Model-Driven Development of Control Software for Distributed Automation : A Survey and an Approach," *IEEE Trans. Syst. Man, Cybern. Syst.*, vol. 44, no. 3, pp. 292–305, 2014.

[32] P. Fishwick, "WHAT IS SIMULATION?," *University of Florida*, 1995. [Online]. Available: https://www.cise.ufl.edu/~fishwick/introsim/node1.html. [Accessed: 27-Dec-2016].

[33] J. Tominaga, M. Ferreira, and J. Silva, "A rule-based satellite simulator for use in flight operations planning," *J. Comput. Interdiscip. Sci.*, vol. 2, no. 2, pp. 111–121, 2011.

[34] Center for Nonproliferation Studies, "Treaty on principles Governing the Activities on States in the Exploration and Use of Outer Space, Including the Moon and Other Celestial Bodies," *Nuclear Threat Initiative*, 2008. [Online]. Available: http://untreaty.un.org/cod/avl/pdf/ha/tos/tos_e.pdf. [Accessed: 27-Dec-2016].

[35] W. Brissett, "Autonomy Should Be Fully Integrated, Carefully Limited," *Air Force Magazine*, 2016. [Online]. Available: http://www.airforcemag.com/DRArchive/Pages/2016/October 2016/October 31 2016/Autonomy-Should-Be-Fully-Integrated,-Carefully-Limited.aspx. [Accessed: 27-Dec-2016].

[36] R. P. J. C. Bose, W. M. P. Van Der Aalst, I. Zliobaite, and M. Pechenizkiy, "Dealing with concept drifts in process mining," *IEEE Trans. Neural Networks*

Learn. Syst., vol. 25, no. 1, pp. 154–171, 2014.

[37] W. M. P. Van Der Aalst, *Process Mining Data Science in Action*, Second. New York: Springer, 2016.

[38] "PropCube 1, 2, 3 (Flora, Fauna, Merryweather)," *Gunter's Space Page*, 2016. [Online]. Available: http://space.skyrocket.de/doc_sdat/propcube-1.htm. [Accessed: 04-Jan-2017].

[39] K. Weiss, T. M. Khoshgoftaar, and D. Wang, *A survey of transfer learning*, vol. 3, no. 1. Springer International Publishing, 2016.

[40] K. Weiss, T. M. Khoshgoftaar, and D. Wang, *A survey of transfer learning*, vol. 3, no. 1. Springer International Publishing, 2016.

[41] L. Valerio, A. Passarella, and M. Conti, "Accuracy vs . traffic trade-off of Learning IoT Data Patterns at the Edge with Hypothesis Transfer Learning," 2016.

[42] Z. Ding, S. Member, Y. Fu, and S. Member, "Robust Transfer Metric Learning for Image Classification," vol. 26, no. 2, pp. 660–670, 2017.

[43] Object Management Group, "Satellite Operations Language Metamodel (SOLM)," no. November, 2012.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
23-03-2017	Master's Thesis	Sept 2015 – March 2017

4. TITLE AND SUBTITLE

Enabling Air Force Satellite Ground System Automation Through Software Engineering

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Bentley, Michael J, 2dLt, USAF

5d. PROJECT NUMBER
JON 17G409, JON 16G904

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
Wright-Patterson AFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Space and Missile Systems Center
Nancy Holt, GG-14
483 N Aviation Blvd,
El Segundo, CA 90245
Nancy.holt.2@us.af.mil, DSN: 263-8437

10. SPONSOR/MONITOR'S ACRONYM(S)
SMC/ADG

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
Distribution Statement A. Approved for Public Release; Distribution Unlimited

13. SUPPLEMENTARY NOTES
This work is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

US Air Force satellite ground stations require significant manpower to operate due to their fragmented legacy architectures. To improve operating efficiencies, the Air Force seeks to incorporate automation into routine satellite operations. Interaction with autonomous systems includes not only daily operations, but also the development, maintainability, and the extensibility of such systems. This thesis researches challenges to Air Force satellite automation: 1) existing architecture of legacy systems, 2) space segment diversity, and 3) unclear definition and scoping of the term, "automation." Using a qualitative case study approach, we survey comparable non-satellite operation domains (Industrial Control Automation and Software Testing) that have successfully integrated automation and other satellite operation enterprises (NASA Goddard, Naval Research Laboratory, European Ground Station National Institute for Space Research in Brazil) to identify common themes and best practices. From this insight, we recommend that future satellite operation ground stations encourage the use of layered architectures, abstract satellite operation processes, and integrate simulators in future systems as concrete implementations of this common operating platform. Further elaborating on the value of these recommendations, this thesis researches the benefits of process mining in satellite operations. This research is conducted on FalconSAT-3 and PropCube 1 and 3. The process mining analysis discovered that a highly structured Concept of Operations (CONOPS) is required in order to gain significant benefits from process mining.

15. SUBJECT TERMS
Software engineering, automation, satellite, ground station, process mining, transfer learning, case study analysis

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Maj Alan C. Lin, AFIT/ENG
U	U	U	UU	95	**19b. TELEPHONE NUMBER** *(include area code)* (937)255-3636 x4757 Alan.lin@afit.edu