DYNAMIC LOGICAL MISSION MODELING TOOL

THESIS

Justin A. Sadowski, Captain, USAF

AFIT-GSS-ENY-17-M-290

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

AFIT-GSS-ENY-17-M-290

DYNAMIC LOGICAL MISSION MODELING TOOL

THESIS

Presented to the Faculty

Department of Aeronautics and Astronautics

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 Space Systems

Justin A. Sadowski, BS

Captain, USAF

March 2017

AFIT-GSS-ENY-17-M-290

DYNAMIC LOGICAL MISSION MODELING TOOL

Justin A. Sadowski, BS

Captain, USAF

Approved:

_____	_____
Eric D. Swenson, PhD, AFIT (Chairman)	Date
_____	_____
Richard G. Cobb, PhD, AFIT (Member)	Date
_____	_____
Bradley J. Ayres, PhD, Contractor (Member)	Date

Abstract

The ability to test and evaluate spacecraft designs is limited by the challenges of getting to (and operating in) the mission environment, and thus modeling and simulation is one way the space industry drives down risk and assists development of new technologies. The more accurate these models and simulations become, the more useful they are to the designer. While there are many choices for orbital propagation software, there are not many that allow dynamic modeling of both the spacecraft (to include its mode states) and its interactions with the environment in which it operates. The *environmental model* includes the spacecraft's orbit and spatial relationship to other agents in the simulation as well as non-agent entities such as planets and stars. The in-house AFIT modeling and simulation software, the Logic-Based Mission Modeling Tool (LMMT), has introduced the capability of behavior-based modeling by defining the spacecraft's modes as logical states in a state machine, however, in its current form it does not allow for changes the satellite or other entities may make in how they interact with the environment in which it operates.

This thesis research evaluates the usefulness of dynamic modeling as compared with static modeling (such as that which is already possible with the LMMT). When changes occur which make the spacecraft's method of interacting with the environmental model no longer relevant, due to either spacecraft mode changes or other agents in the simulation, the model should be dynamically updated to include these changes by means of repropagating the environmental model.

This research's focus is to ideate and then evaluate a subset of use-cases that would create changes in the environmental model. Through this research, it is hoped to develop a method for identifying when a static model such as the LMMT should be utilized versus a dynamic model such as that developed specifically for this research.

Acknowledgments

I would like to express my gratitude for my research advisor, Dr. Eric Swenson, for his unflagging support, infectious energy, and optimistic outlook about what can be accomplished in the world. It is through his example that the truth that one can never be too filled with wonder is made abundantly clear. I would also like to thank the members of my committee for working through this document with me. A special note goes out to Chaplain Snyder, who sponsored a productivity class that served to guide my efforts more efficiently. Finally, the rest of my fellow students deserve credit for pushing me to fully realize the project that this has become and supporting me when things got unnecessarily exciting toward the end.

Justin A. Sadowski

Table of Contents

Page

List of Figures

List of Tables

DYNAMIC LOGICAL MISSION MODELING TOOL

I Introduction

1.1 Overview

The challenges of operating a spacecraft in its design environment and the barriers to entering space itself have always made spacecraft design a fertile ground for constructing and evaluating models and simulations, and it is this modeling and simulation effort that this research focuses on. As is stated in *Simulation Modeling Handbook: A Practical Approach* [1], "Simulation modeling and analysis is the process of creating and experimenting with a computerized mathematical model of a physical system." Given the cost of testing inside the operational space environment alongside the difficulties of replicating the same conditions on Earth (cost-effectively or otherwise), modeling and simulation is a valuable addition to most missions. Currently, however, commercial orbital propagation software available to space professionals has limited or no capability to handle logical decision-making effects. This research focuses on considering the possibilities afforded by software which allows such logical decision-making to have direct impacts upon the *environmental model* (as defined later).

Very few missions today utilize spacecraft that are only ever in one mode of operation. And, as may be intuitive, the more accurate a model is, the more useful and pertinent the predictions produced will be and thus the more-informed decisions based off those predictions will be. As discussed above, commercial orbital propagation software has limitations regarding its ability to include logical decision-making capabilities within the simulations themselves. Software such as Analytical Graphics, Inc.'s Systems

ToolKit (STK) [2], The Aerospace Corporation's Satellite Orbit Analysis Program [3], or code written in MATLAB does not typically provide a comprehensive method to evaluate an on-orbit spacecraft as it proceeds through differing modes of operation. Therefore, in the course of this research, a feedback mechanism is developed which will allow a model to change how the spacecraft interacts with the environment and vice versa over the course of a simulation, with the goal of better reflecting the actual operation of a spacecraft. This dynamic approach to modeling can then be utilized to evaluate whether a dynamic or static approach is better suited to the chosen use-cases as they exhibit expected spacecraft behaviors.

By allowing a model to logically decide how it will respond to certain stimuli or triggers in the environment in a dynamic fashion (e.g. when a UAV enters an airspace, a targeting radar switches from search mode to target track mode) extra information can be gained from the simulation about the suitability of the chosen concept of operations (CONOPS) for the system. In the actual operational life of a system, responses to the environment are not made in a void where nothing in the environment responds back or changes. The architecture built by three prior AFIT students [4–6] as a result of their thesis research serves as a solid foundation for further research into this area, and the focus of this research is to ideate and then evaluate a subset of use-cases that create changes in the environmental model. This research will first build a new capability onto the existing ones provided by the Logic-Based Mission Modeling Tool (LMMT) [6]; a feedback mechanism which can drive an 'environmental repropagation' or a regeneration of the data provided by the environmental model (as defined in the following section). Following this, a non-inclusive list of events and/or actions by an agent which would

have an impact on how the spacecraft interacts with the environmental model during a simulated spacecraft mission are conceived. A subset of these events and/or actions is chosen to be replicated in use-cases to be evaluated by the software, and conclusions are drawn determining what sort of modeling projects would benefit from a dynamic approach as opposed to the simpler static approach.

1.2 Definitions

To add some clarity to this research endeavor, a brief glossary of key terms used and their meaning in the context of this research follows. Throughout this thesis, where any term's definition may be in question, italicization will denote that the definition listed below is to be used. These definitions are not by any means considered standard or widely-used throughout the modeling world. *Environmental Model* in particular is simply a chosen set of words to reflect a concept that was otherwise not easily explained in a clear and concise manner.

Mode: a state of operation of a system, specifying certain actions or responses to stimuli (e.g. during Survival, the spacecraft's attitude and determination control subsystem (ADACS) will be powered down and the spacecraft will tumble). Jewell [5] took a different approach and delineated a difference between the 'state' of a system and 'mode,' which is deemed unnecessary for this particular research's purposes. 'Mode state' is sometimes used to illustrate this blending.

State Machine: a logic-based decision-making set of triggers and responses that allows an *agent's* behavior to be regulated by a series of rules derived from the desired concept of operations (CONOPS), operating in discrete time steps

Agent: a system or subsystem whose actions or interactions could be governed by a state machine (e.g. a satellite, a subsystem, a ground station)

Environmental Model: the model that replicates the physical world in order to govern how *agents* interact within a simulation (e.g. orbit propagator, RF link suite, weather model), to be held as distinct from the model of the system, such as the satellite or other *agent*

1.3 Background

This section begins by providing the reader more depth about modeling and simulation, further developing this research's validity of purpose. Next, an overview of prior AFIT students' research into this area, with an emphasis on the software this research is built upon is provided. The section concludes with a high-level discussion of what a state machine is and how it applies to modeling spacecraft behavior. This provides the reader a better understanding of the type of models which form the logical decision-making segment of the software this research focuses on.

1.3.1 Modeling and Simulation

Modeling and simulation, as a tool for risk reduction and design validation, has been heavily leveraged by the space community because of the operationally challenging environment most spacecraft face as well as the high cost of entry into that environment (i.e. cost of launch) [7]. Beyond the expense of the materials and electronics suitable for the space environment, launch costs run in the thousands per kilogram. SpaceX's Falcon 9 [8] can launch to geostationary transfer orbit at ~$11,300 per kilogram while the United Launch Alliance's Atlas V [9] can launch to the same orbit at ~$18,600 per kilogram.

This price gets even more expensive when measured not by pure, amorphous mass but by mass and volume, which is a more relevant/realistic measure of launch cost. When measured in units of CubeSat volume (with one *U* equal to a 10x10x10cm cube [10]), one 12U, 20 kg spacecraft costs nearly $3M to launch into the same geostationary transfer orbit [11], and this is while using a ride-share program to save on costs. Clearly, there are severe entry barriers to getting into space.

According to *Space Mission Engineering* [7], modeling is "the art and science of creating virtual builds (or, models) of systems of varying complexity and levels of integration." Simulation then refers to evaluating how a model changes/interacts over time. The text continues, "Today's space systems are designed and deployed in the context of existing surface-based and space-based systems. Interactions within the subject mission and with these existing systems should be exercised early and often…" Furthermore, "It is rarely practical to exercise an end item or test article in the full mission environment and durations before committing it to space deployment and fielded service." As discussed at the beginning of this subsection, to even get into actual space incurs a program severe cost penalties, which apply whether the spacecraft is intended for operational use or just for testing. But even once there, the spacecraft is subjected to a variety of hardships not easily replicable on earth (i.e. vacuum, thermal loading, radiation, etc.). As a result, models are used to gain insight into predicted behavior of designs and stretch limited resources further. Given that modeling and simulation are firmly entrenched in the space industry, improving the fidelity of these models is an area of continuing research, including this research.

1.3.2 AFIT Student Efforts Overview

Students of the Air Force Institute of Technology (AFIT) have been designing and refining a suite of spacecraft-specific modeling tools written in MATLAB since 2012 [12]. The latest three of these produced the pieces of and then the integrated software suite of the LMMT [4–6]. These efforts have resulted in a software suite that leverages the simulation capabilities of Simulink, the analytical and display capabilities of MATLAB proper, and the logic-based decisions capability afforded by Stateflow. Most recently, Loudermilk [6] combined the logical decision capabilities of the CubeSat State Analysis Tool (CSAT) [5] with the analytical processing of the Mission Modeling Tool (MMT) [4]. Driven by an initial (and importantly, static) set of data reports from STK, the integration of both tools into a single software suite allowed Loudermilk to pave the way for further research into mode-based CubeSat mission analysis. More information on the initial CSAT and MMT can be found later in this thesis, in their respective developers' theses, and, most recently, in Loudermilk's thesis, *A Logic-Based Mission Modeling Tool for Designing CubeSats* [6]. More details about the lineage of these efforts is found in Section 2.2 and Table 1.

1.3.3 Finite State Machine

This subsection draws on knowledge and definitions provided in the *Encyclopedia of Computer Science and Technology* [13]. Finite-State Machines (or more simply, state machines) are ways of looking at a system as a black box with a discrete number of input and output ports. This concept applies whether the system being modeled is defined as software, hardware, or some sort of hybrid. In respect to spacecraft modeling, these

systems are the modes in which a spacecraft exists. The action of the system (the response to inputs) is constrained to occur at discrete time-steps, which is important to note when constructing these models in environments such as Stateflow/Simulink, which can only process actions in discrete time steps. State machines, therefore, fit well into this constraint/design consideration. While not as "strong" as the Turing model of computing, state machines prove useful in modeling the behavior of systems that react to outside stimuli in a consistent and pre-planned fashion. Simply stated, the "strength" of a computational model is a measure of its ability to solve problems. Turing models can be described as abstract computational models which are the most capable; able to solve any problem [that is solvable by means of an algorithm].

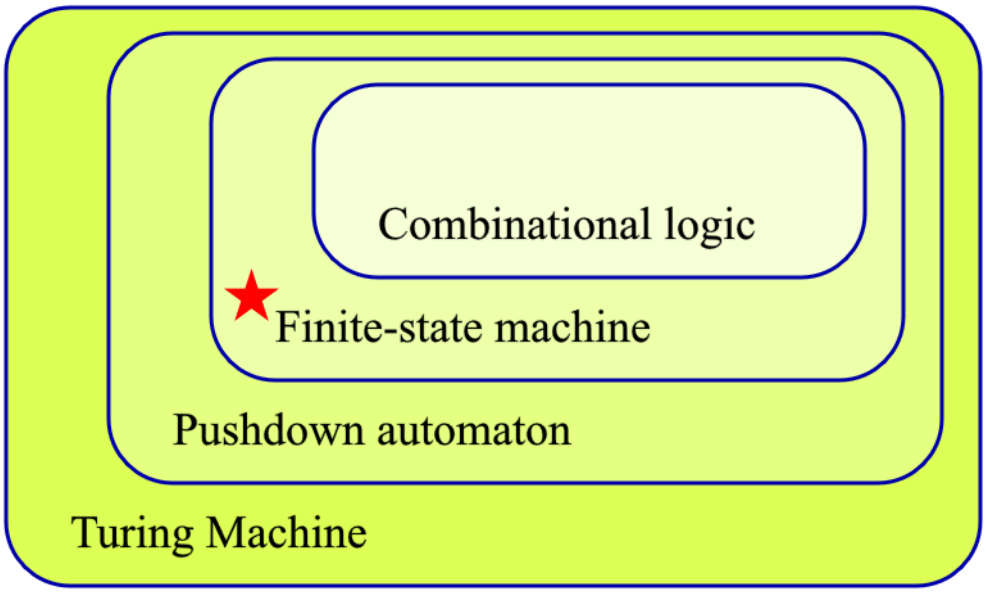

Figure 1: Relative Complexity of Classes of Automata – Further-Out Layers are More Complex/More Capable [14]

Modeling spacecraft, however, allows the use of the lower-tier (as relative to the graphic in Fig. 1) state machine class as there are indeed only a finite series of states the spacecraft can be in. State machines can be used to represent the spacecraft in its entirety

or can be nested in layers to represent individual subsystems; each different state defined by the logic it uses to process the inputs and outputs it receives and produces. In this research, the spacecraft's modes are represented as states, with triggers leading to entry and exit from each state and actions accomplished while in that specific mode state. For example, the spacecraft will enter SunSafe mode state when the battery depth of discharge reaches a set, user-input, level. State machines accurately and efficiently model the behavior of a spacecraft which operates within a predefined set of modes.

1.4 Problem Statement

This thesis research evaluates the usefulness of dynamic modeling as compared with static modeling (such as that which is already possible with the LMMT). When changes occur that make the spacecraft's interactions with the *environmental model* no longer relevant, due either to the spacecraft's actions (such as spacecraft maneuvers) or that of other *agents* in the simulation, the *environmental model* should be updated and repropagated to account for these changes. This research's focus is to ideate actions and events that would create changes in the spacecraft's interactions with the *environmental model* and then evaluate a subset of these actions in use-cases via a software which includes the capability to repropagate the *environmental model*. Through this research, it is intended to develop a method for identifying when a static model such as the LMMT should be utilized versus a dynamic model such as that developed specifically for this research.

1.5 Methodology

1.5.1 Research Objective

The research's objective is to develop a method of identifying when a dynamic modeling approach would provide more value than a simpler static approach to spacecraft modeling. Included in this objective is the development of software that will add the capability to dynamically model and simulate a spacecraft over a given mission, repropagating the *environmental model* autonomously when necessary. This software will allow use-cases to be run to illustrate some of the possible actions or events which would invite repropagation. The LMMT does not (by design) allow for any changes to its *environmental model* data as the original desire during its inception was to develop a 'fast' simulation which still included the orbital propagation and behavior [12]. It does, however, form a starting point from which additional repropagations can be introduced into the model.

1.5.2 Scope, Assumptions, and Limitations

There are three primary limiting factors in this research.

- The decision-making that goes into what change in the satellite model is significant enough to justify repropagation will not be investigated in this research. It should be left up to the individual spacecraft designer to determine what necessitates repropagation and when said repropagation should occur. For this research, a set of mode changes deemed 'significant' is selected and implemented.

- Although more than just spacecraft mode changes may give impetus to repropagate the environmental model (i.e. mobile ground stations, moving targets) this research will not provide them the separate state machines that would be necessary to allow them to drive repropagations, nor will this research include them in the environmental model.
- The calculations made within the framework of the MMT 2015 (the MATLAB post-processing work) have not been validated [6], and this research does not seek to do so. Whatever values provided to the state machine side of the LMMT are assumed to have been derived correctly for the purposes of this research.

1.6 Thesis Overview

This thesis follows a five-chapter format. This section concludes Chapter 1: Introduction and Background. A review of the relevant literature and others' research is provided in Chapter 2 in order to frame the research in its appropriate context, and Chapter 3 discusses the methodology and means by which the research was conducted. Chapter 4 provides the reader with the results of the use-cases chosen to demonstrate the new software's functionality and discusses their behavioral repercussions of certain design choices. Chapter 5 presents the larger significance of the results in Chapter 4, discusses remaining work, suggests future research, and concludes the thesis.

II Literature Review

2.1 Chapter Overview

First, this chapter will discuss the software this research is built upon, the LMMT, to include a more in-depth look at the two progenitors of the LMMT and a discussion of similar software capabilities provided by a commercial modeling plug-in to STK. The chapter concludes with a discussion of AGI's AMOEBA plug-in, which seeks to utilize state machines and a commercial simulation tool kit to accomplish many of the same aims as this research. The AMOEBA initiative, which was not available for evaluation and testing at the time of this research, was a driving consideration in developing the focus of this research.

2.2 Foundational Software

This section details the LMMT and the two immediately-preceding efforts that led to its creation. Additionally, a table is provided below cataloging the complete efforts of the AFIT ENY spacecraft modeling and simulation effort as begun in 2006 with Project Insight [15]. Table 1 presents a list of the previous thesis work on the AFIT modeling efforts starting from 2006. Efforts are generally listed in chronological order when applicable – when multiple theses were completed in the same year, they are listed alphabetically with respect to their author.

Table 1: Timeline of AFIT Spacecraft Modeling Efforts

Thesis/Effort	Primary Author/Researcher	Year
Project Insight: Threat Modeling And Assessment For Earth-Orbiting Satellites [15]	Reed M. Bond et al.	2006
A Simulink Based Tool for Design Reference Mission Modeling [16]	Jusdon E. McCarty	2010
A Colony II CubeSat Mission Modeling Tool [17]	Blythe Andrews	2012
Electrospray Propulsion Interface and Mission Modeling for CubeSats [18]	Angela Hatch	2012
Applying Model-Based Systems Engineering to CubeSats: A Tailored Approach for a Reusable State Analysis Tool [5]	Benjamin A. Jewell	2015
A CubeSat Mission Modeling Tool [4]	Heather M. Udell	2015
A Logic-Based Mission Modeling Tool for Designing CubeSats [6]	Joshua R. Loudermilk	2016

2.2.1 The Mission Modeling Tool 2015

Udell [4] upgraded the Mission Modeling Tool from the uncompiled standalone computer-based research code that it was into a MATLAB-based classroom-ready tool for students to analyze CubeSat designs and architectures. Originally written with the Colony II bus (as designed by Boeing) in mind, Udell updated the MMT to remove many of the elements that were hard-coded to specifics of the Colony II bus, allowing for a wider range of CubeSat designs to be evaluated. The MMT 2015 would take a set of STK scenario data and post-process it, generating plots detailing various telemetry data points of interest to the modeler and demonstrating if the chosen design met the chosen mission parameters. Modes were not so much switched between as triggered on and off based on values that were set by the modeler. For instance, the bus voltage could be set such that

when a certain value was passed, the MMT would report the satellite in ‘Survival Mode’. No changes to either the satellite or the *environmental model* would occur, and the simulation would continue logging data. Should the voltage rise above the threshold level again, the satellite would be logged as no longer existing in Survival Mode. As Udell wrote, “In an actual mission, the modeler would not want the satellite to eternally switch between these two modes. Instead, they would want the satellite to stay in one mode or the other long enough to recharge the batteries, reset, or otherwise correct the issue. However, the MMT is designed merely to indicate potentially detrimental orbit conditions or improperly sized hardware early in the design process. *Any* triggering of sun safe, and especially of survival mode, signifies that there is a problem with the design that needs to be resolved.” The intent of the MMT 2015 was not to evaluate CONOPS choices or the response of the satellite in a hierarchical mode structure. In Loudermilk’s modification, the MMT 2015 essentially became the trigger generator for the CSAT state machine, replacing the task list and adding further realism to the simulation.

While STK proper does not allow for the sort of calculations the MMT 2015 provides to the LMMT, a plug-in is available called SOLIS [19]. This third-party software from Advanced Solutions Inc. allows a long list of modeling and simulation techniques to be run inside the STK environment, as detailed in the list on their website [19]. For example, under the Attitude Determination Modeling, the plug-in touts its ability to model sun sensors, horizon sensors, rate sensors, magnetometers, and star trackers with options for perfect attitude determination, fixed-gain filters, and/or Kalman filters being applied.

SOLIS is a relatively new plug-in, first introduced in 2011 [20], and new features are being added as they are coded and tested, so in a few years it is planned to be even more capable. However, even with the added capability, SOLIS will not be suitable for the focus of this research, as it does not support STK Connect scripting (as of this writing). Due to the autonomous nature of the desired feedback loop, some sort of script-based initialization is a required feature.

2.2.2 CubeSat State Analysis Tool (CSAT)

Jewell's thesis [5] led to the state machine controller of the LMMT. Jewell created a working model of a satellite bus and its operations based on a representative CubeSat mission within Stateflow/Simulink, and the CSAT was the result. Applying Model-Based Systems Engineering principles, Jewell created a tool that provided "reusable states, modes, and logic in a CubeSat modeling framework." One drawback that Jewell had to contend with was that by running the state machine as a standalone simulation, the operator had to inject change events and triggers manually into a 'task list' to validate how the logical models interacted. Loudermilk, when integrating the CSAT into the MMT architecture, was able to do away with manual triggering by setting up the MMT to provide the necessary injects to the state machine at each discrete step throughout the simulation.

2.3 The Logic-based Mission Modeling Tool

Because this research is built upon the work done to produce the LMMT, a description of its evolution is provided in this section and its subsections. In Swartwout's article [21], it is noted that the vast majority of the failed CubeSat missions were

university-led projects, and, more tellingly, nearly half of all university-led CubeSat missions (even of just those that reach orbit) fail to achieve mission success. The information provided in this 2013 article was cited as a driving consideration for several of the student research efforts into developing software that could be used at government locations (to include AFIT) to help design and model CubeSats. The latest generation of these efforts is the LMMT; designed and coded by Loudermilk [6] in the Spring of 2015. The LMMT allows a student to test a specified physical design against a basic mission as outlined in an STK scenario that is run prior to the simulation. Given the inherent simplicity of the LMMT framework, it could serve to help size major components for initial design and a first-pass look at requirements. However, because the LMMT only requires one run of STK and its reports, it cannot incorporate and propagate any changes in the *environmental model* (as discussed in Section 1). This research seeks to evaluate if this can be changed.

2.3.1 LMMT: High-level Operation

Loudermilk's [6] LMMT ingests data reports generated by STK (with the reports being run prior to the simulation) at the beginning of each simulation run and then references the information at each time step along the simulation. This data remains static; following the initial ingest nothing more is done with the *environmental model* data files. The content of these data files includes the sun and moon vectors, ground station access, target-in-view windows, and the attitude and ephemeris of the spacecraft. These files are the only means by which the spacecraft model can 'see' the outside world. Once the initial generation scenario has been run in STK and the requisite reports generated, the

environmental model data is treated as unchanging – referenced but never regenerated. The focus of this research is to test the feasibility of a method that would allow this data to remain relevant by providing the capability to repropagate (where needed) during the simulation itself. Since the LMMT was initially two pieces of software that were integrated together (the MMT 2015 [4] and the CSAT [5]), this thesis will refer to one portion or the other as separate elements, even though they are fully integrated and work seamlessly together to produce the outputs from LMMT. The former CSAT can be thought of as the logical "brain" of LMMT, and will henceforth in this thesis be referred to as the logical side or the state machine. The former MMT 2015 can be thought of as a series of calculations that are run at each time step (which, for this research, is set to 60 seconds) utilizing the *environmental model* data, preparing higher-level values such as battery depth of discharge from said raw data and modeler inputs detailing the spacecraft's physical characteristics. Loudermilk, while integrating the MMT 2015 and CSAT, highlighted modeler input locations in blue in the software (see Fig. 2 as an example), and this convention continues to be utilized in this research.

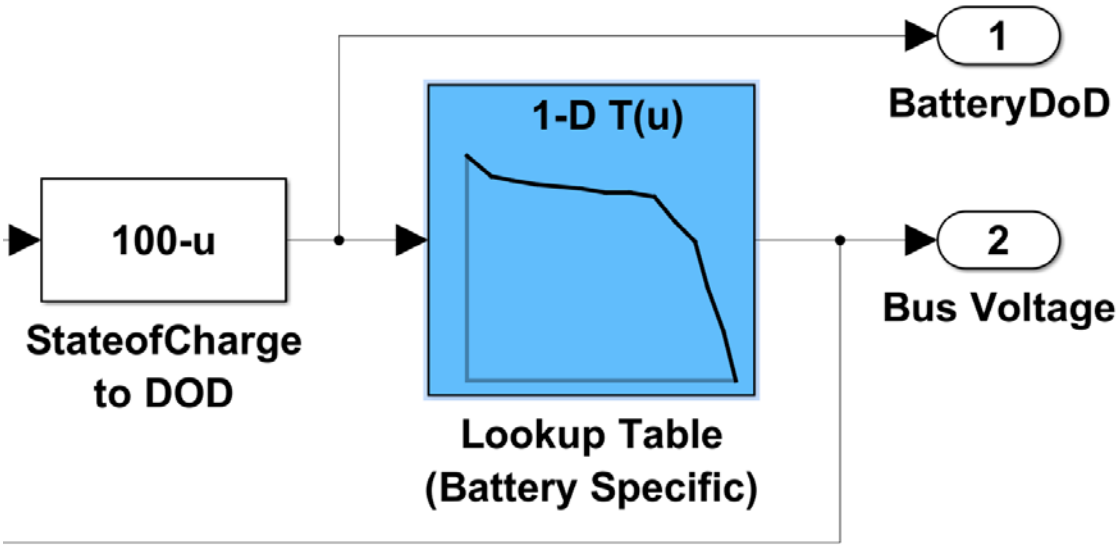

Figure 2: Close-up of Blue Modeler-Input Block in EPS, DyLoMMT (Simulink screenshot)

These modeler inputs are what allow the modification of the software to fit new/different designs and provides a flexibility that greatly aids in the design and development process. Also of note is that the LMMT software is designed to be opened, values for the spacecraft entered, and then a simulation run of the model – in that order. The *environmental model* creation and data report generation have to be accomplished prior to running the MATLAB/Simulink simulation, in a related but unconnected process.

2.3.2 LMMT: Pre-Simulation Initialization Stage

Before any simulation run can be started in LMMT, the modeler first generates a mission-representative *environmental model*. Loudermilk provided an excellent user's guide [22] describing how to use the STK graphical user interface (GUI) to generate basic scenarios and, even more helpfully, how to produce the specific reports the LMMT relies upon for its *environmental model* data. After the reports are generated correctly (formatted to meet the specific guidelines described in the *How to Use Manual* [22]), the

modeler places them into the folder the LMMT is expecting, and processes them (by means of provided code) into a format recognizable by the rest of the software. The modeler then opens (as opposed to runs) the LMMT Simulink model and adjusts the physical parameters of the spacecraft inside the model.

Once both the *environmental model* and physical model have been input into the LMMT (the former through creation and processing of reports, the latter by direct entry into the LMMT model in Simulink), the modeler can now actually run the LMMT simulation – the logical state machine operations and concurrent data calculations.

2.3.3 LMMT: Logical Processing

The logical "brain" of the LMMT is a series of interconnected states diagrammed in a Stateflow chart as seen in Fig. 3 and fed through the Simulink simulation in accordance with the wiring as seen in Fig. 4. It is this state machine (Fig. 3) that decides which mode the spacecraft should exist in at each time step, as defined by its logic (as it evaluates the higher-level values provided for it by the ongoing calculations). The calculations are generated by means of various Simulink block diagrams and the occasional MATLAB function block when more complicated mathematical operations are desired. These calculations are leveraged again in the DyLoMMT; mostly in their original format, as discussed in the research limitations (Section 1.5.2).

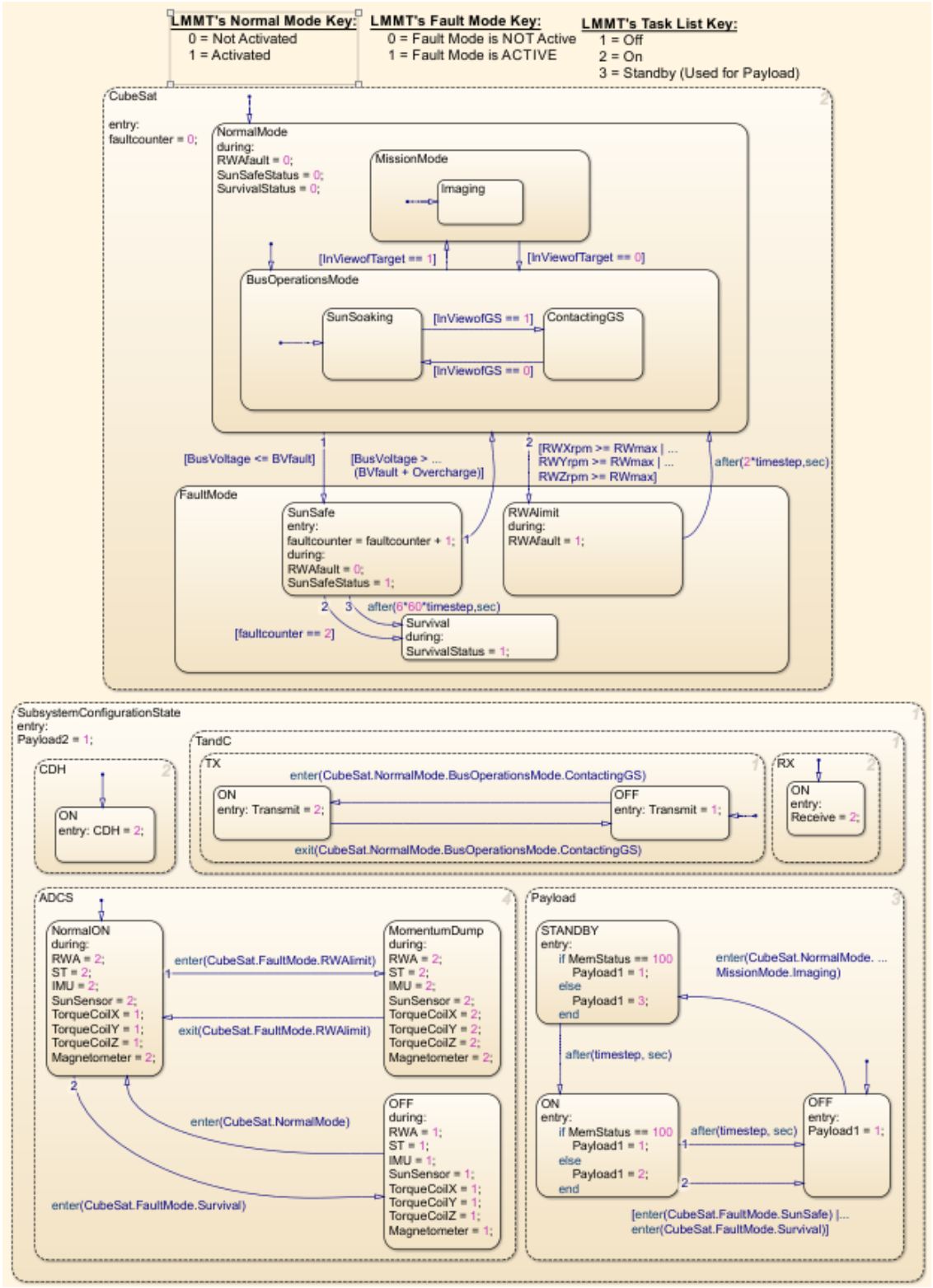

Figure 3: Full State Machine Diagram, LMMT

(Stateflow screenshot)

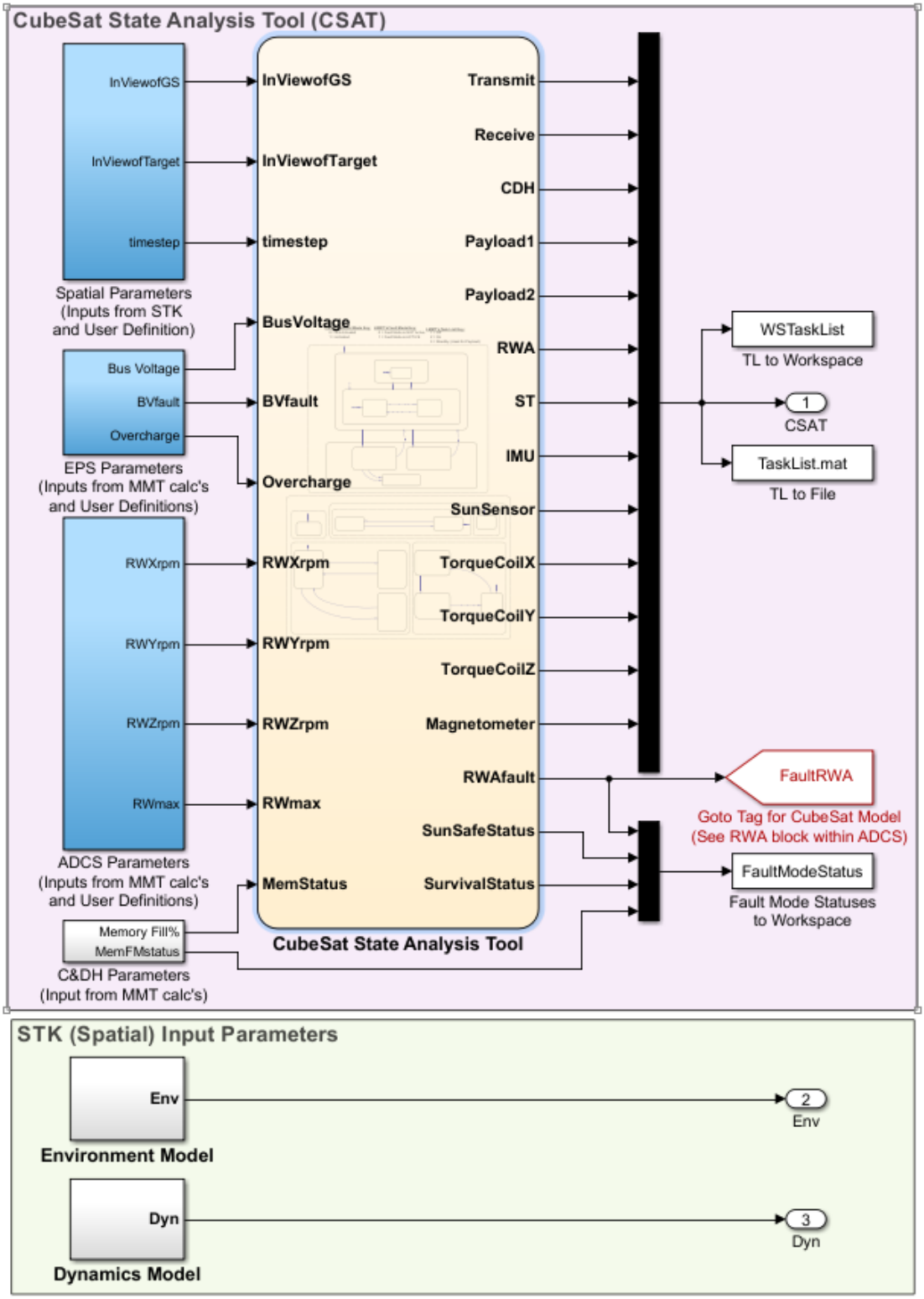

Figure 4: Wiring Diagram of State Machine, LMMT

(Simulink screenshot)

2.4 AGI's AMEOBA Plug-in

Haun [23] (of AGI) has conceptualized and then led the development of a plug-in for STK that allows the *environmental model* to interface with a No Magic product called Cameo Systems Modeler [24], running its Cameo Simulation Toolkit [25]. "Cameo Systems Modeler ™ is an industry leading cross-platform collaborative Model-Based Systems Engineering (MBSE) environment, which provides smart, robust, and intuitive tools to define, track, and visualize all aspects of systems in the most standard-compliant SysML models and diagrams." The Cameo Simulation Toolkit plug-in plays a role in allowing the models to be run within the Cameo Systems Modeler – all of which gives AMEOBA the state machine functionality that Stateflow provides the LMMT.

The AMOEBA plug-in functions by means of a series of state machines built inside Cameo Systems Modeler which control their respective *agent's* behavior inside the STK environmental simulation. This constant feedback system is accomplished with the assistance of the AMOEBA plug-in's Information Layer, which allows the Cameo state machines to be built using a language such that environmental stimuli or triggers will be read into Cameo from STK and commands from the various *agent's* state machines will be translated back into commands that STK can interpret and act upon. The Information Layer serves as a library of terms that can be used to port information back and forth between Cameo's state machines and the STK *environmental model* simulation, allowing modelers not fluent in Java programming to use the same interface without having to code their own 'translators'. This effort will allow Cameo and STK to communicate more seamlessly, much like the methods already developed for the MATLAB/STK connection [26] that are leveraged in this research.

2.5 Other State-Machine-Driven Software Suites

As of this writing, no other commercially-available satellite design suites that are driven by state machines have been found. It is thought, however, that many major satellite development programs utilize some level of state behavioral analysis during their design and test phase, but due to the proprietary nature of many satellite designs, their approaches are not openly advertised. What has been discovered is presented below.

Some university satellite design programs discuss using state machines to model the behavior of their spacecraft or subsystems therein. For example, a NASA PowerPoint presentation [27] describes a CubeSat systems engineering example (that of the AubieSat-1 from Auburn University's Student Space Program) where state machines are suggested as a modeling technique to assist the designers in the down-selection and optimization processes.

In their book *Solar Tracking*, Prinsloo and Dobson [28] suggest automating solar tracking control systems with state machines, which would correspondingly be easily modeled by the same. Furthermore, model checking methods work by representing the software as state machines [29], and this task is made easier if the model is already represented as such.

The presumption that state machines are indeed a useful tool for modeling spacecraft is supported by Kaslow's [30] work with the International Council of Systems Engineering's (INCOSE) Space Systems Working Group. The group has set out a path toward providing a CubeSat reference model in which state diagrams are used to model "behavior in response to internal and external events." The trend is indeed toward

utilizing state machine modeling in conjunction with *environmental models* to replicate on-orbit behaviors.

Given these examples, it is presumed that state machines will continue to be used to model the behaviors of spacecraft in academia and industry. The next step is seen as introducing the ability to dynamically model these states and any changes they may introduce to the *environmental model*; this research seeks to identify when this new approach should be considered.

2.6 Summary

This chapter discussed the foundational efforts that the research software developed here was based upon in order to provide a background on the previous work in the area of spacecraft modeling and simulation. Extra information was provided on the immediate predecessor to the DyLoMMT (the LMMT) as it provides much of the functionality of the new software and because its merits are reviewed in the conclusions of this thesis. This chapter also summarized the findings of other software using state machine modeling of spacecraft, most notably the AGI AMEOBA plug-in, whose fully-dynamic approach to modeling provided the impetus to engage upon this research. With ever-increasing options for dynamic modeling available, answering the question of when or in what cases is dynamic modeling desired/required becomes ever more pertinent.

III Methodology

3.1 Chapter Overview

The purpose of this chapter is to discuss how the research objective from Subsection 1.5.1 is addressed: identifying when a dynamic or static modeling approach should be chosen. To achieve this objective, it is necessary to develop software which will allow the capability of using either a dynamic or static approach to model spacecraft behaviors, and this effort will be briefly discussed in this chapter. The chapter begins with sections detailing the two commercial software programs (STK and MATLAB) that this research utilizes to accomplish its goal (Fig. 6). Of note to the reader is that these software programs are needed only in terms of their functionality (i.e. *environmental model* propagation and state machine logic/calculations, respectively). Any other software capable of performing these functions as defined within this research could be used in their stead, and this idea is discussed further at the end of each of software sections. When discussing the software in terms specific to the configuration used in this research, the respective name (e.g. STK) is used. However, when relating to these programs in their more general sense, *state machine* and *environmental model* are used instead with the aim to further aid outside application of this research's presented information. Figure 5 is presented as a top-level visual depiction of the concept used to conduct this research.

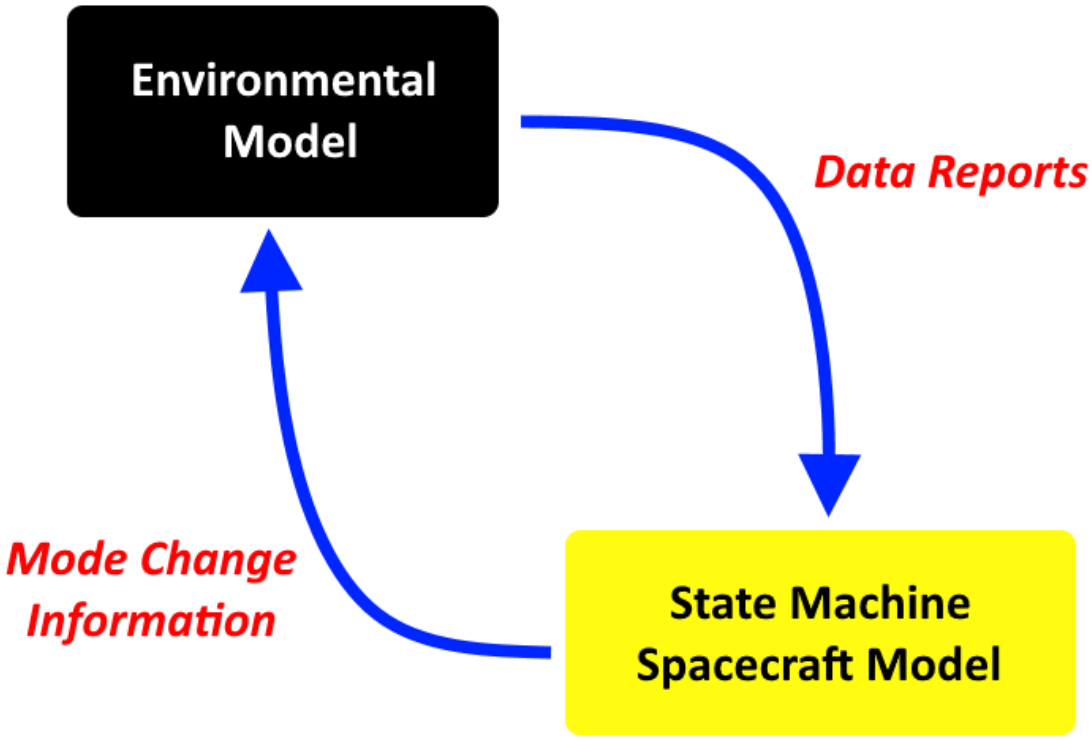

Figure 5: Top-Level Feedback Loop Flowchart

The 'Data Reports' seen in Fig. 5 are the *environmental model* data used to link the two sides of the software together. In the case of the LMMT, this is the series of reports generated by STK mentioned earlier and discussed again in Section 3.2. Significant effort went into determining how (and in what ways) to modify the LMMT in order to support the research objective, but that effort is not discussed in detail in this thesis. The new software developed for this research is called the Dynamic Logical Mission Modeling Tool (DyLoMMT), paying homage to the hard work of the other students whose efforts it is built upon while delineating its signature difference. The chapter concludes with a description of the representative mission and the series of use-cases which are used to achieve the research objective.

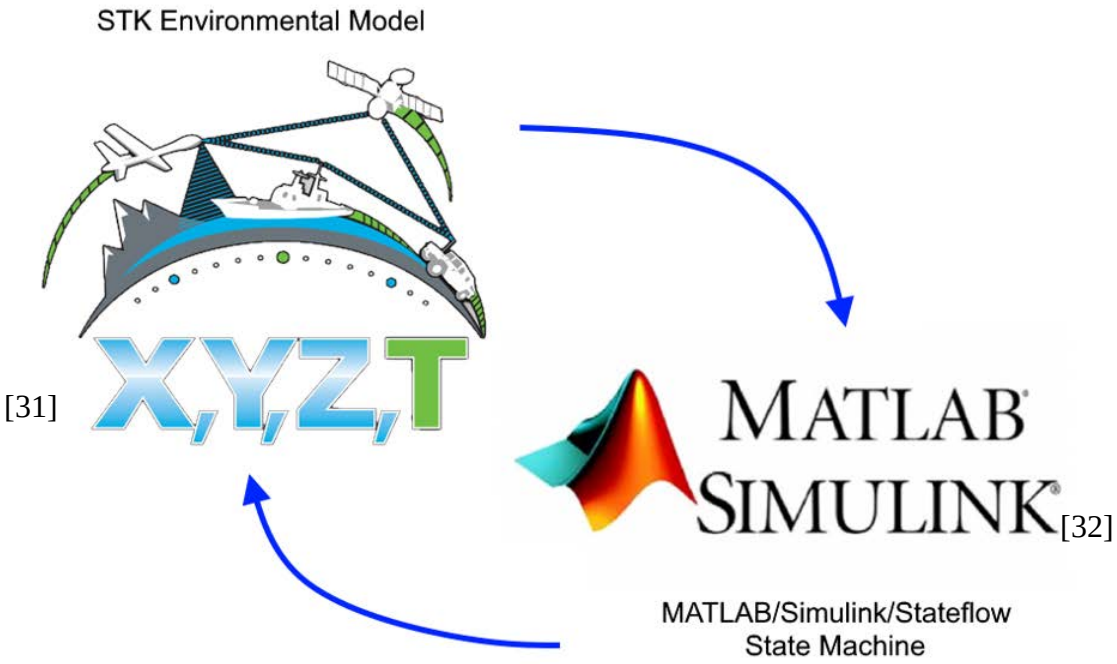

Figure 6: Two Pieces of Software Integrated Together [31] [32]

3.2 STK

Analytical Graphics Inc.'s [33] general mission as a company is to provide software to model, analyze and visualize space, defense, and intelligence systems. Their Systems ToolKit (STK) [2] software serves as the environmental model for the LMMT and plays an even bigger role in this research's work. Billed in AGI's STK flyer [31] as a physics-based modeling, simulation and analysis tool that allows visualization through its integrated 4D (X, Y, Z, and Time) interactive globe, STK is used in this research to generate text-formatted reports about target and ground station access and directional vectors to the Sun and the Moon. These reports are then used by the calculations side of the LMMT to inform logical arguments such as "target in view of payload", or the angle the sun vector forms with the vector normal to the solar arrays for power generation

statistics. Again, the information these reports provide is the only way in which the spacecraft model can ‘see’ what is happening in the environmental model. Finally, while not useful to the simulation itself, the visualization capability of STK allows easier evaluation of the mission scenario for obvious coding errors (e.g. missing spacecraft or incorrectly located ground stations). While STK is the *environmental model* used by the DyLoMMT and its progenitors, any software which produces the same information that the DyLoMMT ingests and is capable of being autonomously controlled via scripting could be utilized in this capacity. The concept remains the same, although the connections and implementation may be completely different.

3.3 MATLAB

While the LMMT uses the series of initial reports from STK (Sun and Moon vectors, access to ground stations and targets, attitude and ephemeris, etc.) described in the preceding section to feed (in one direction) the spacecraft model, the main analytical computation of this modeling and simulation software is completed inside a software suite programmed over the years by several prior students [4–6]. This software suite is written in MATLAB and makes use of products from the Simulink family (both Simulink main and Stateflow).

MathWorks is the developing company for both the MATLAB Product Family [34] and the Simulink Product Family [32] (which includes Stateflow [35]), and both are used in tandem to create the main analysis engine of the LMMT. Each of these continues their functions as such in this research. Self-identified as “the Language of Technical Computing,” MATLAB began as a large matrix manipulator and calculator (with

MATLAB initially standing for MATrix LABratory) and has since grown into its own programming language with a variety of pre-coded capabilities that are provided by MathWorks as add-ons, or toolboxes. While not the most efficient of programming languages, MATLAB's emphasis on data handling and more colloquial syntax make it a favorite among engineers who do not possess other, more traditional, programming skills. Simulink is a block diagram environment that runs inside MATLAB proper and is designed specifically to run finite time step simulations. A feature of Simulink heavily utilized by both the LMMT and the DyLoMMT is Stateflow, which is an environment that allows modeling and simulation of combinatorial and sequential decision logic, all based on state machines and flow charts. Stateflow forms the state machine model portion of the LMMT and continues to play said role in this research.

3.4 Development of the DyLoMMT

As part of this research was the development of a feedback mechanism that would allow dynamic repropagation of the *environmental model*, a brief discussion of the modifications to the LMMT are included in the following subsections.

3.4.1 Adding Autonomy and the Feedback Mechanism

Under the LMMT framework, the STK-generated data reports are collected once at the beginning of the simulation, and the STK scenario to generate them is accomplished by means of STK's graphical user interface. In order for the software to dynamically regenerate the spacecraft's perception of the *environmental model* data, this process needed to be automated. Using the Connect [36] library of commands available from AGI, a script was developed which created a full mission-representative scenario

and produced the necessary reports to initialize the LMMT. This code could then be called from other script files to enable it to be run autonomously from within the simulation. An example of this code is included in the Appendix. A visualization of the DyLoMMT running a simulation is presented in Fig. 7.

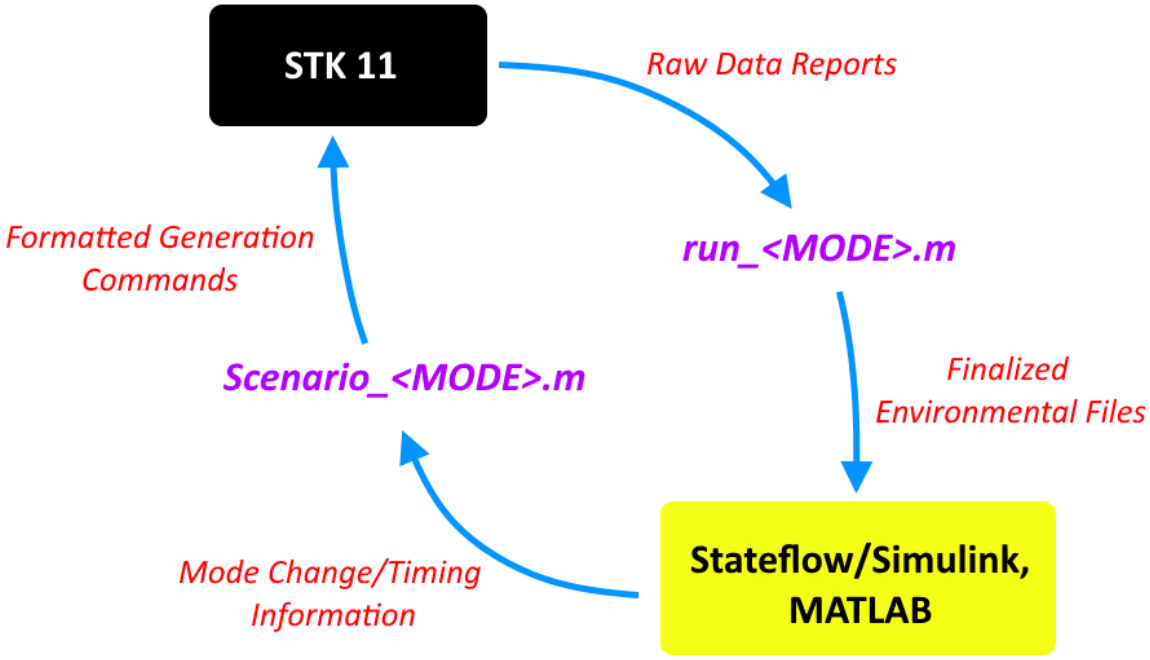

Figure 7: DyLoMMT Simulation Flowchart

These automation scripts then allowed for the logical side of the DyLoMMT to be modified to allow it to call for repropagations as necessary/defined by the state diagram's logic. Stateflow [37], in which the state machine runs, allows outside MATLAB functions to be called by means of the aptly-named function-call block, which itself must be called by one of the operational states (Fig. 8).

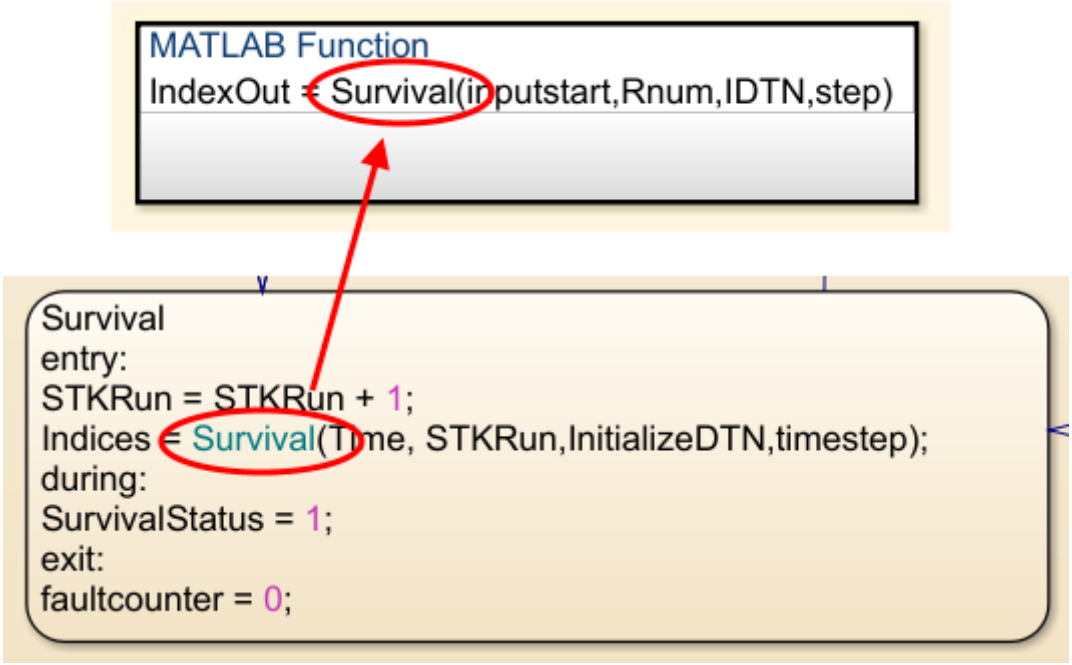

Figure 8: Function Call Block and Calling State, DyLoMMT
(Stateflow screenshots, combined with additions)

Upon the spacecraft model entering a new state/mode, the DyLoMMT calls the related function block and this starts a separate function script inside MATLAB. The Simulink part of the simulation then pauses until the completion of this function call, and then continues to step forward in time. This cycle continues as many times as commanded by the state machine until the simulation reaches the end of the scenario. Neither the details of these scripts, the mechanics of their function within the DyLoMMT, nor their development is provided in this thesis.

3.4.2 The Processing Capabilities of the LMMT and the new DyLoMMT

In between pauses to change the *environmental model* data, the calculation side of the simulation continues in its linear step fashion, determining higher-level values needed by the state machine to evaluate its logic. As this research is focused on the relative

merits of static and dynamic approaches to modeling, the calculations side of the LMMT has been modified only when necessary. The calculations, therefore, run as designed by the previous coders working on the software, with the exception of unavoidable alterations, such as when the initial date and time chosen for the scenario start was required to be passed into the state machine, or when the Survival Overcharge modeler input was added. The spacecraft model in the DyLoMMT is best visualized from the 'dashboard' view provided in Fig. 9, which is where the modeler interacts with the subsystem mask they wish to edit (via double-clicking) and inputs his/her own values for that subsystem. By editing each of the subsystems' parameters in this fashion, the model is 'built' according to those specifications. Currently, the configuration possibilities are limited to those provided in these masks. Adding new possibilities requires further alteration of the Simulink code/blocks underneath the mask. The other modeler-editable values are those related to the state machine's operation and presented via the state diagram chart, as discussed previously. Of note are the ports leading into and out of the dashboard view block (seen at the bottom of Fig. 9); these ports/tags are explained in more detail in Loudermilk's thesis [6]. Since the repropagation scripts edit the main *environmental model* files and overwrite new data during a pause in the simulation, the calculations proceed as they did in the LMMT, time step by time step, pausing as the state machine pauses.

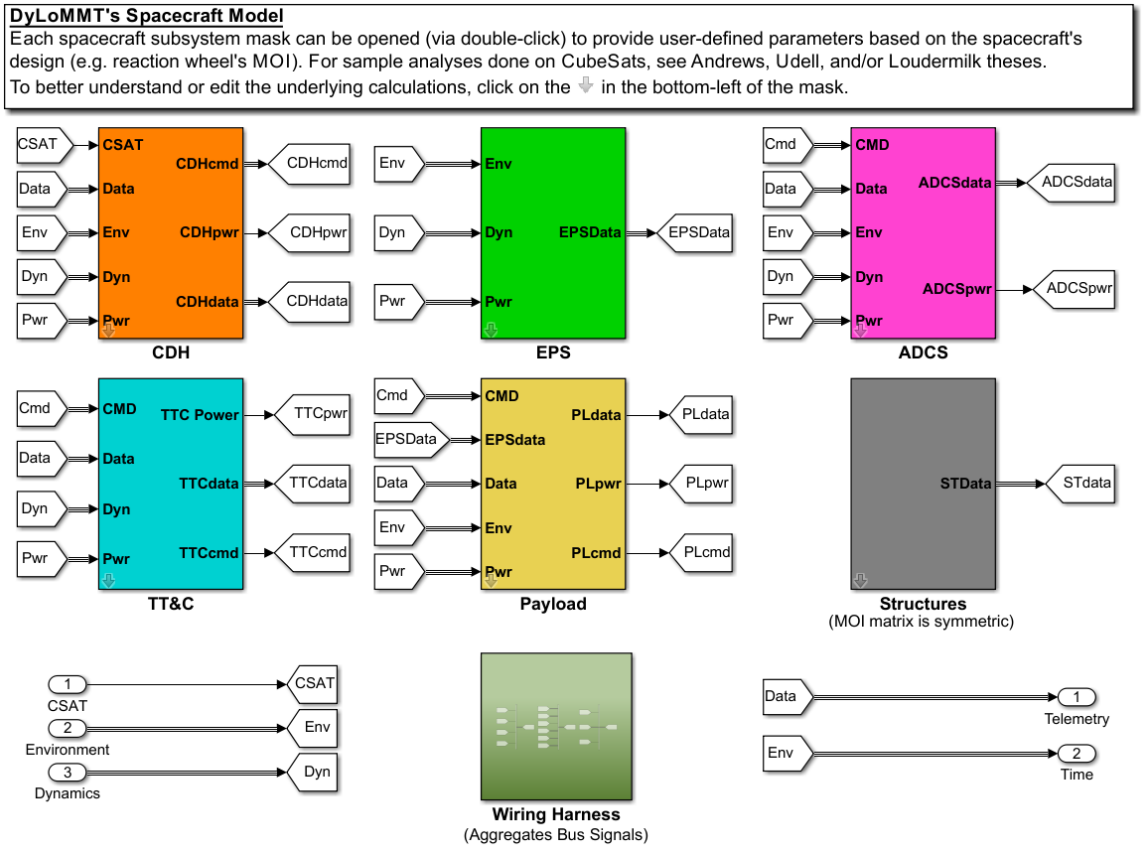

Figure 9: Dashboard view, DyLoMMT

(Simulink screenshot)

3.5 Additional Changes Made

The changes made to the LMMT described above, while of main import to this research as they enable the dynamic modeling approach, are not the only changes made to the LMMT. These next changes had to do with the operation of the model inside the simulation, and were guided by lessons learned from discussions in the ASYS 632 [38] satellite design course offered at AFIT. First of note is that the mode transition from Mission to either SunSafe or Survival is now triggered by the *Depth of Discharge* of the battery pack (as simulated) instead of by the *Bus Voltage*. This replaces the state transition that used to be triggered by the bus voltage reaching a certain, modeler-defined

level. Now, instead of a critical bus voltage to reach, the modeler is asked to input a depth of discharge at which SunSafe will be triggered as well as a recharge percentage required before returning to Mission mode (the *Overcharge* value). Inside the SunSafe mode, the Sun Soaking/Contacting Ground Station loop is added as the spacecraft should be able to communicate with the ground station(s) even while in this fault mode. The TandC configuration state was also modified to account for this change. Finally, the Payload configuration state loop is updated in the following manner. The initial attempt of modification now has the payload begin in the Off configuration, but transition to the Standby state after a period of ten time steps has passed. The Payload then remains in Standby until the spacecraft registers that it is in range of a target. Upon this triggering, it proceeds to the On state. It remains in the On state (unless the spacecraft enters a fault mode) until the spacecraft no longer has access to (a) target(s), at which time it returns to Standby and remains powered. The Payload then remains in the Standby state until the next imaging opportunity (or until a fault mode is entered), as is shown in Fig. 10. In good practice, no subsystem should be powered down completely in normal operations once it was successfully initialized [38], as sometimes components and/or entire subsystems don’t come back on again if power is toggled. Further significant changes to the state machine of the LMMT are discussed in their applicable use-cases.

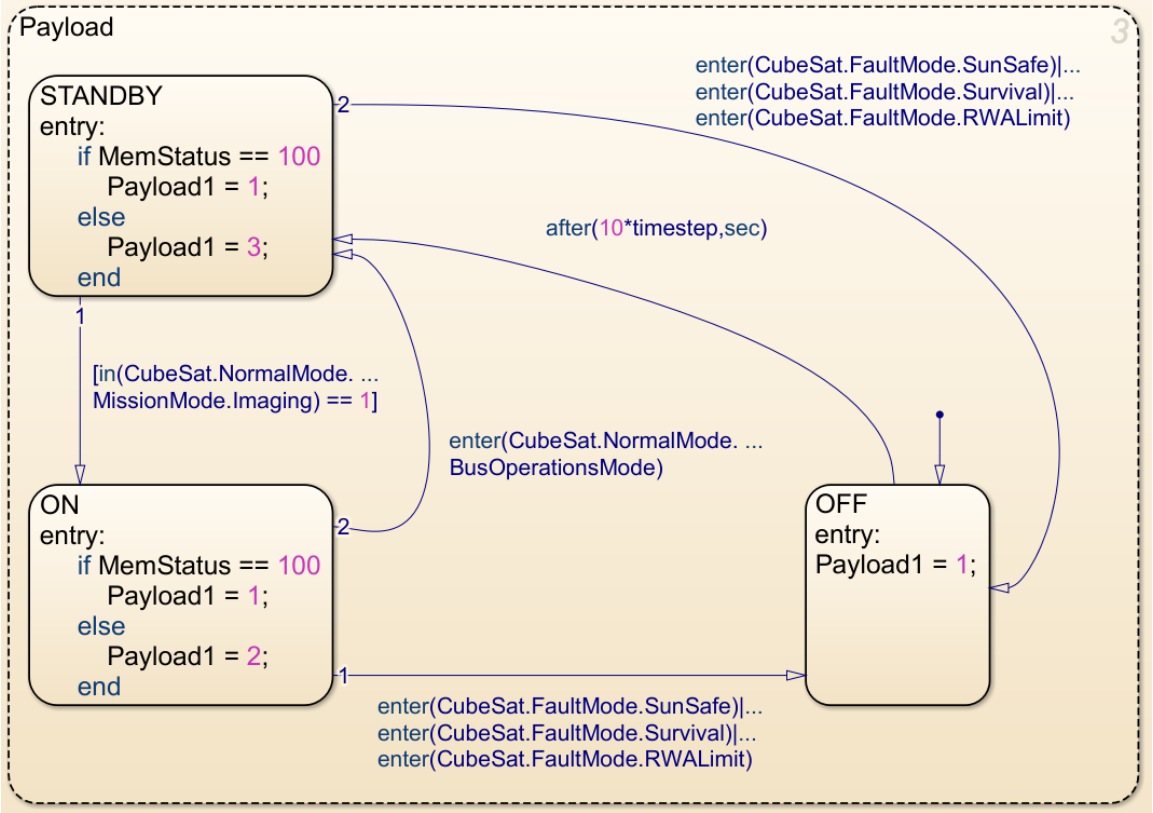

Figure 10: Payload Configuration State, DyLoMMT
(Stateflow screenshot)

3.6 Concept of Operations for Representative Mission

To use the DyLoMMT, a mission representative scenario was chosen. It was decided that a simple imaging mission where a single-payload imaging spacecraft would be launched into Low Earth Orbit (LEO) to monitor a series of target locations scattered around the world would provide opportunities to examine both static and dynamic approaches. A notional orbit was defined, and the scenario was coded into the *.m* files that control the initial and repropagation runs of the *environmental model*. These parameters are hard-coded into the scripts the DyLoMMT uses to function, however, the initial *Scenario_Creation.m* file is the model from which the other modes' scripts are

generated, and would be the place to start when designing in a new mission to be evaluated.

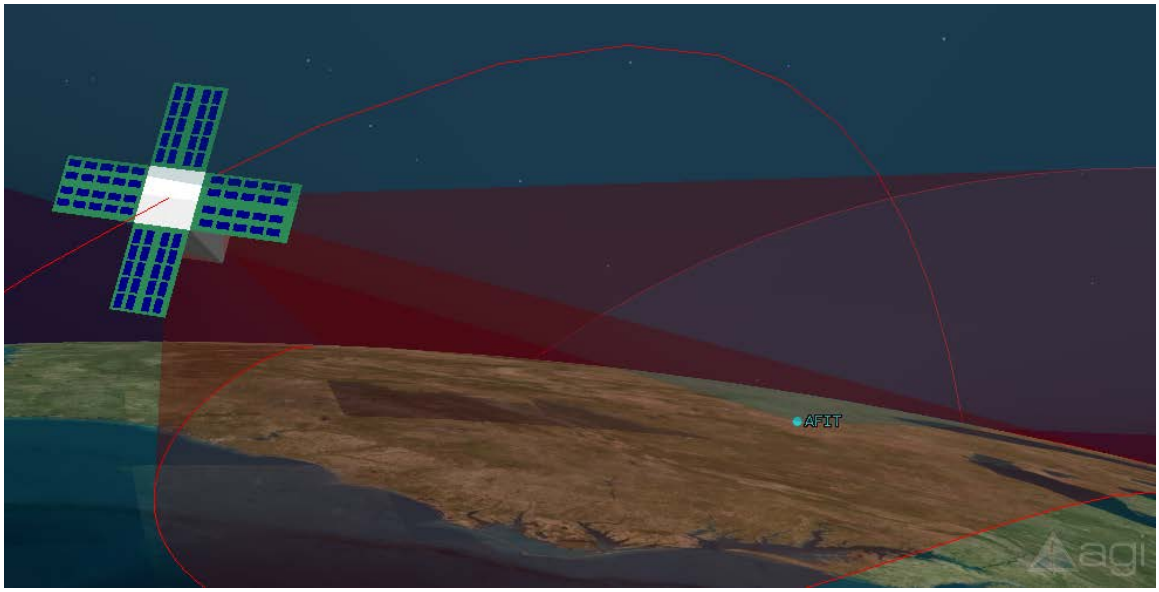

Figure 11: Notional LEOSat 1 Passing Over AFIT Ground Station

3.7 Use-Cases

A series of use-cases designed to help answer the question of when a static vs dynamic approach is best applied is now presented to the reader. Each use-case is run once with its designed repropagations, and then again with these repropagations disabled to provide the comparison of static vs dynamic modeling. The scenario length remains set at 10 days and the *environmental model* initial generation script are not modified from use-case to use-case to provide more easily comparable results. However, the modeler-input values and the state diagrams for each use-case do generally vary and the differences are highlighted to the reader in each subsection.

3.7.1 Generalized Imaging Mission: Use-Case #1 (UC1)

Using the values derived from [39] and the CONOPS specified in the state diagram as shown in Fig. 12, this case shows a baseline for the new software, reassuring the reader that the same functionality exists in the DyLoMMT that was available in the LMMT. These parameters are meant to be realistic values, and are not contrived for the purposes of illustrating the software's functionality.

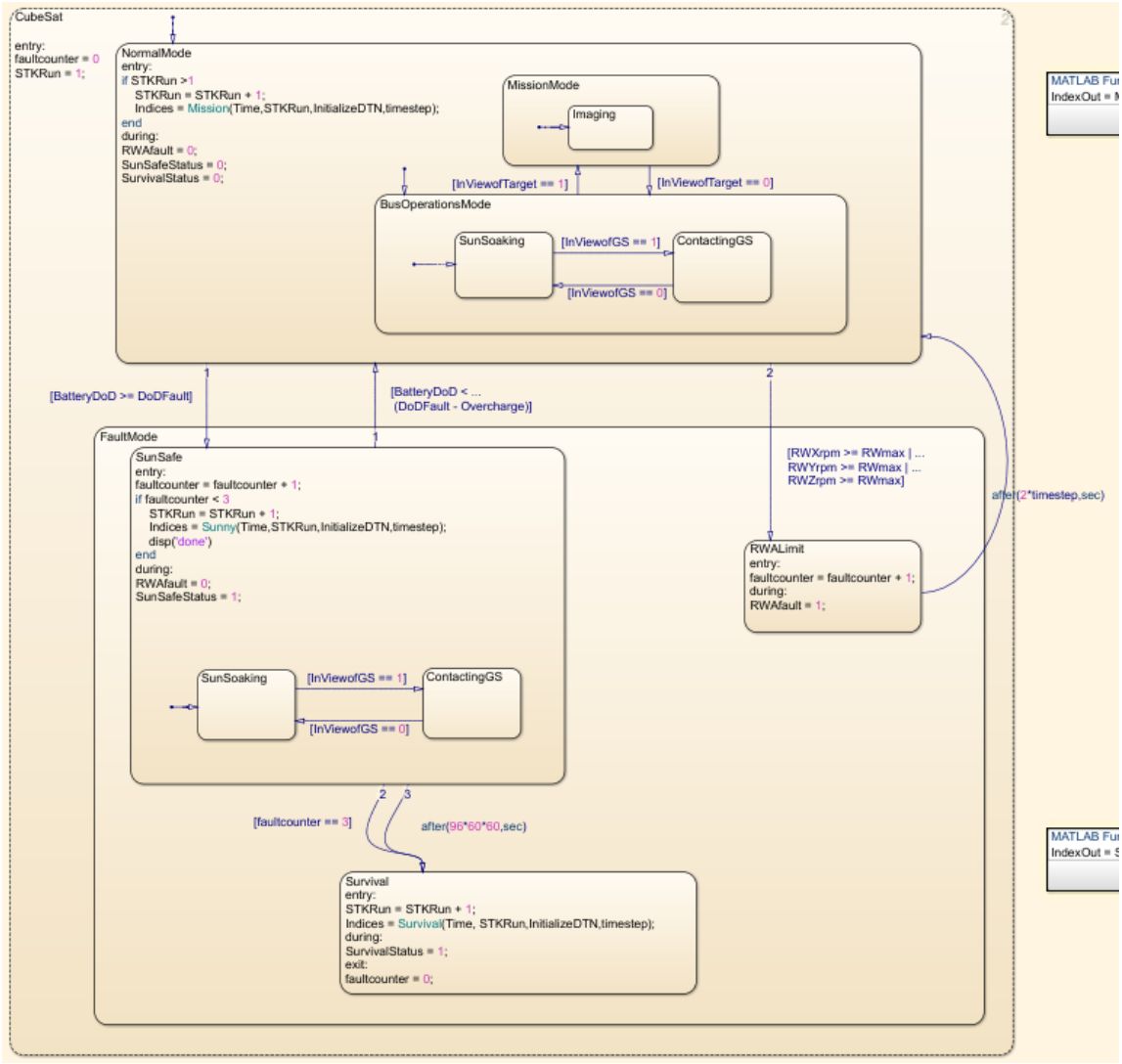

Figure 12: UC1 State Diagram, DyLoMMT
(Stateflow screenshot)

Table 2: UC1 – CD&H and TT&C Parameters

CD&H	
Bus SoH Data Rate	10 bytes/sample time
Payload SoH Data Rate	10 bytes/sample time
Extra Data Rate	0 bytes/sample time
Memory Full Status	10 MB
CDH On Power	3.5 W
Memory Initial Status	0 bytes

TT&C	
Rx: Power On	0.7 W
Rx: Standby Power	0.7 W
Tx: Power On	16 W
Downlink Data Rate	8000 (kbps)

Table 3: UC1 – EPS and Payload Parameters

EPS	
Battery Initial Fill	70 Amp-Hours
Battery Capacity	80 Amp-Hours
Solar Panel Area	0.2 m^2
Solar Panel Efficiency	0.23 (%/100)
Solar Panel Voltage	24 V
Depth of Discharge	50%
Overcharge	15%

Payload(s)	
Payload 1: Power On	27 W
P1: Operating Data Rate	20 kilobytes/sample time
P1: Standby Power	9 W
P1: Standby Data Rate	0 bytes/sample time
Payload 2: Power On	0 W
P2: Operating Data Rate	0 bytes/sample time
P2: Standby Power	0 W
P2: Standby Data Rate	0 bytes/sample time

Table 4: UC1 – ADCS and Structures Parameters

ADCS	
Magnetometer: Power On	1 W
Star Tracker: Power On	1 W
Sun Sensor: Power On	1 W
Torque Coil X: Power On	2 W
Torque Coil Y: Power On	2 W
Torque Coil Z: Power On	2 W
Reaction Wheels: Max Power	5.5 W
RW: Zero-torque Power	1.12 W
RW: Max Momentum	0.015 N-m-s
RW: Moment of Inertia	2.1e-5 kg-m^2
RW: Initial RPM	0 rpm

Structures	
Ixx MOI	0.188 kg-m^2
Ixy MOI	0 kg-m^2
Ixz MOI	0 kg-m^2
Iyy	0.183 kg-m^2
Iyz	0 kg-m^2
Izz	0.009968 kg-m^2
Center of Mass X	0.122 m
Center of Mass Y	0.110 m
Center of Mass Z	0.183 m
Total Mass	10.9 kg

3.7.2 Straight into Fault: Use-Case #2 (UC2)

The next use-case demonstrates a spacecraft designed such that it cannot support mission operations for the entirety of the representative scenario, and proceeds rapidly from Mission mode, through SunSafe, and into Survival. This use-case shows the ability of the DyLoMMT to cycle through different modes in simulation and generate additional *environmental model* runs, as well as appending the main data files used by the simulation with the newly-updated information specific to that mode. For ease of comparison, the altered parameters have been highlighted in their respective tables. No change has been made to the state diagram from UC1.

Table 5: UC2 – CD&H and TT&C Parameters

CD&H	
Bus SoH Data Rate	10 bytes/sample time
Payload SoH Data Rate	10 bytes/sample time
Extra Data Rate	0 bytes/sample time
Memory Full Status	10 MB
CDH On Power	3.5 W
Memory Initial Status	0 bytes

TT&C	
Rx: Power On	0.7 W
Rx: Standby Power	0.7 W
Tx: Power On	16 W
Downlink Data Rate	8000 (kbps)

Table 6: UC2 – EPS and Payload Parameters

EPS	
Battery Initial Fill	70 Amp-Hours
Battery Capacity	80 Amp-Hours
Solar Panel Area	0.2 m^2
Solar Panel Efficiency	0.11 (%/100)
Solar Panel Voltage	24 V
Depth of Discharge	50%
Overcharge	15%

Payload(s)	
Payload 1: Power On	40 W
P1: Operating Data Rate	20 kilobytes/sample time
P1: Standby Power	15 W
P1: Standby Data Rate	0 bytes/sample time
Payload 2: Power On	0 W
P2: Operating Data Rate	0 bytes/sample time
P2: Standby Power	0 W
P2: Standby Data Rate	0 bytes/sample time

Table 7: UC2 – ADCS and Structures Parameters

ADCS	
Magnetometer: Power On	1 W
Star Tracker: Power On	1 W
Sun Sensor: Power On	1 W
Torque Coil X: Power On	2 W
Torque Coil Y: Power On	2 W
Torque Coil Z: Power On	2 W
Reaction Wheels: Max Power	5.5 W
RW: Zero-torque Power	1.12 W
RW: Max Momentum	0.015 N-m-s
RW: Moment of Inertia	2.1e-5 kg-m^2
RW: Initial RPM	0 rpm

Structures	
Ixx MOI	0.188 kg-m^2
Ixy MOI	0 kg-m^2
Ixz MOI	0 kg-m^2
Iyy	0.183 kg-m^2
Iyz	0 kg-m^2
Izz	0.009968 kg-m^2
Center of Mass X	0.122 m
Center of Mass Y	0.110 m
Center of Mass Z	0.183 m
Total Mass	10.9 kg

3.7.3 Into and Out of SunSoak, Then Survival: Use-Case #3 (UC3)

This use-case examines the design of a spacecraft that, under these mission parameters, slips into SunSafe mode, but is able to recharge its batteries to the point where it returns to Mission mode. However, upon reenabling the payload (and associated power draw) the spacecraft immediately drains its batteries past their fault depth of discharge yet again, which this time sends the spacecraft into Survival. This immediate move from Mission mode into Survival is brought about by a change in the state diagram, changing the option to enter Sun Safe mode twice during the simulation before being forced into Survival to only allowing one Sun Safe occurrence before the next depth of discharge fault sends the spacecraft into Survival. Figure 13 gives a close-up of the SunSafe mode and its triggers that send it to Survival. The *faultcounter* logic statement is altered to trigger at 2, as opposed to 3 from the previous use-cases. The *if* statement found in the entry conditions for SunSafe is necessary to avoid beginning a repropagation run for SunSafe while conditions have been met to skip to triggering Survival (and its associated repropagation run). Otherwise, the runs will conflict as the scripts attempt to

use the same information port to communicate with STK. Thus, the *if* statement skips over any SunSafe repropagation when the conditions for Survival are met, and care must be taken to ensure it remains in agreement with the entry to Survival condition. Finally, it should be noted that this case does not allow for an autonomous recovery from Survival mode, nor do the previous two cases. In the tables following, the values changed from the immediately preceding use-case are again highlighted in their respective tables.

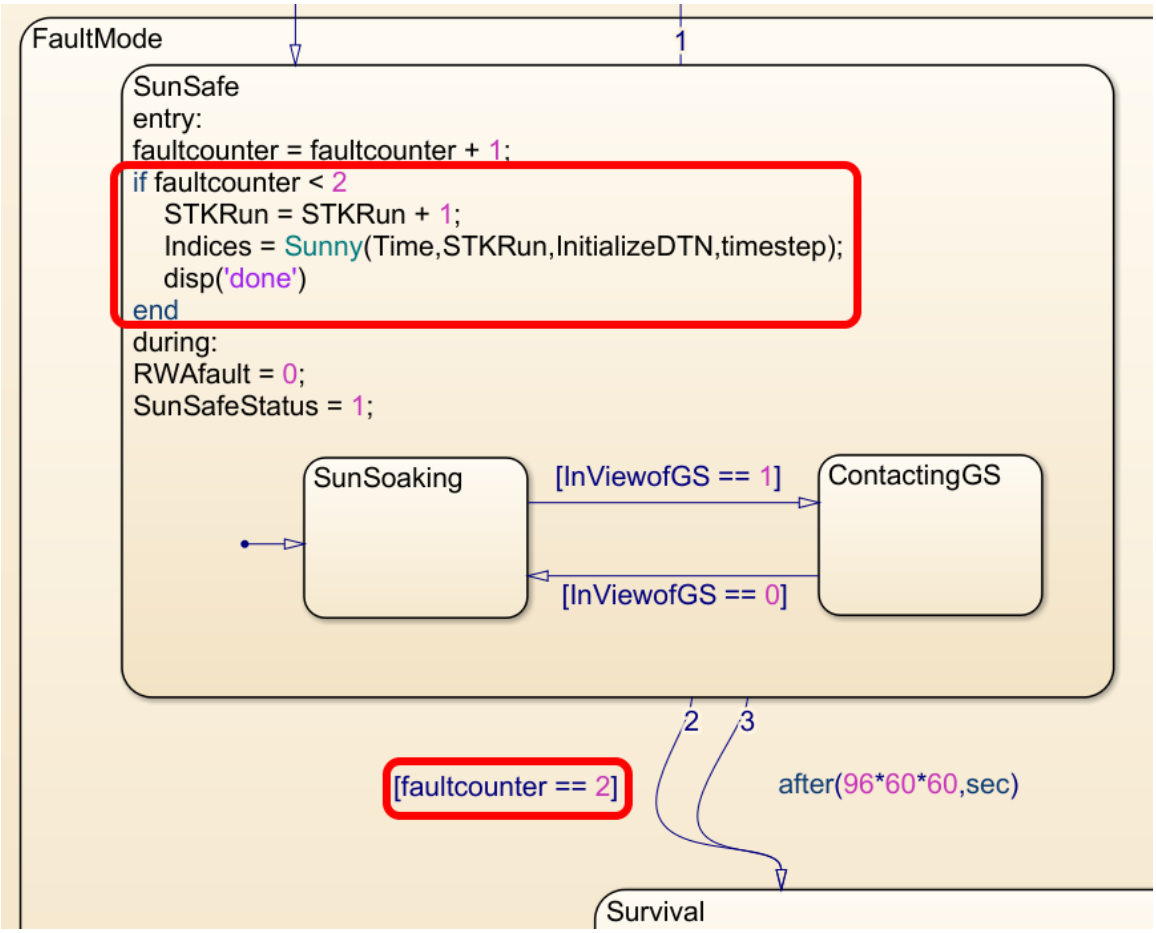

Figure 13: UC3 State Diagram, DyLoMMT

(Stateflow screenshot, with additions)

Table 8: UC3 – CD&H and TT&C Parameters

CD&H	
Bus SoH Data Rate	10 bytes/sample time
Payload SoH Data Rate	10 bytes/sample time
Extra Data Rate	0 bytes/sample time
Memory Full Status	10 MB
CDH On Power	3.5 W
Memory Initial Status	0 bytes

TT&C	
Rx: Power On	0.7 W
Rx: Standby Power	0.7 W
Tx: Power On	16 W
Downlink Data Rate	8000 (kbps)

Table 9: UC3 – EPS and Payload Parameters

EPS	
Battery Initial Fill	70 Amp-Hours
Battery Capacity	80 Amp-Hours
Solar Panel Area	0.2 m^2
Solar Panel Efficiency	0.17 (%/100)
Solar Panel Voltage	24 V
Depth of Discharge	50%
Overcharge	15%

Payload(s)	
Payload 1: Power On	40 W
P1: Operating Data Rate	20 kilobytes/sample time
P1: Standby Power	15 W
P1: Standby Data Rate	0 bytes/sample time
Payload 2: Power On	0 W
P2: Operating Data Rate	0 bytes/sample time
P2: Standby Power	0 W
P2: Standby Data Rate	0 bytes/sample time

Table 10: UC3 – ADCS and Structures Parameters

ADCS	
Magnetometer: Power On	1 W
Star Tracker: Power On	1 W
Sun Sensor: Power On	1 W
Torque Coil X: Power On	2 W
Torque Coil Y: Power On	2 W
Torque Coil Z: Power On	2 W
Reaction Wheels: Max Power	5.5 W
RW: Zero-torque Power	1.12 W
RW: Max Momentum	0.015 N-m-s
RW: Moment of Inertia	2.1e-5 kg-m^2
RW: Initial RPM	0 rpm

Structures	
Ixx MOI	0.188 kg-m^2
Ixy MOI	0 kg-m^2
Ixz MOI	0 kg-m^2
Iyy	0.183 kg-m^2
Iyz	0 kg-m^2
Izz	0.009968 kg-m^2
Center of Mass X	0.122 m
Center of Mass Y	0.110 m
Center of Mass Z	0.183 m
Total Mass	10.9 kg

3.7.4 Into and Out of SunSafe Multiple Times: Use-Case #4 (UC4)

This case demonstrates a situation where the spacecraft oscillates between the Mission and SunSafe modes, never settling into Survival. To accomplish this, the depth of discharge recharge level required to reenter Mission mode is halved to 7% and the state diagram is modified (as is illustrated in Fig. 14) to allow for repeated triggers of SunSafe, without dropping into Survival. Note that in this case the trigger in question is not removed entirely because the same result can be accomplished when the count of needed cycles before heading straight to Survival is artificially placed at a level high enough (arbitrarily chosen to be 20) to be out of reach for the simulation. Again, care is needed to ensure the *if* statement's conditions match up with the corresponding entry conditions as boxed in red in Fig. 14. While this method is simpler, for a longer simulation for which the same behavior is desired, it is recommended that this counter/trigger be removed entirely (leaving only the time as the sole SunSafe trigger). This will require the 'if' statement inside the SunSafe mode to be also altered correspondingly to ensure the repropagation would run correctly.

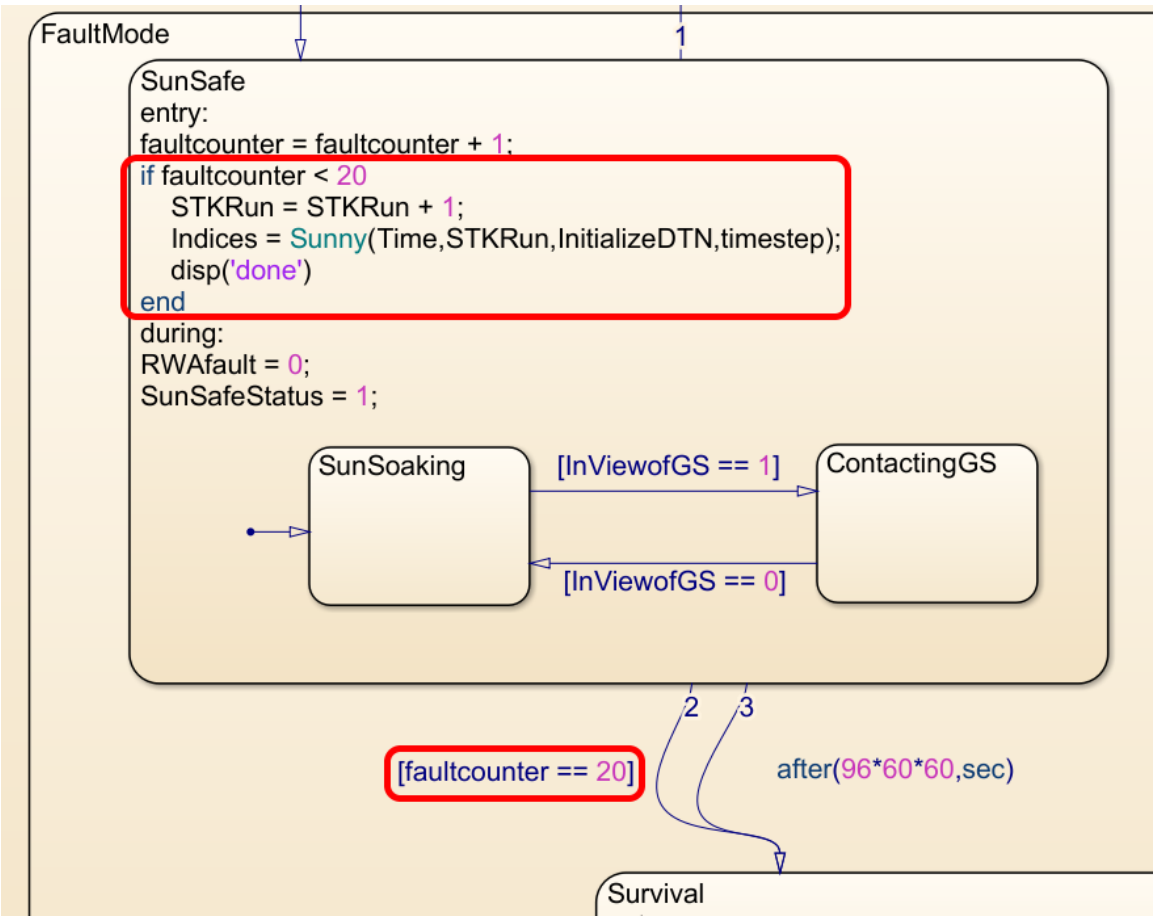

Figure 14: UC4 State Diagram, DyLoMMT

(Stateflow screenshot, with additions)

Table 11: UC4 – CD&H and TT&C Parameters

CD&H	
Bus SoH Data Rate	10 bytes/sample time
Payload SoH Data Rate	10 bytes/sample time
Extra Data Rate	0 bytes/sample time
Memory Full Status	10 MB
CDH On Power	3.5 W
Memory Initial Status	0 bytes

TT&C	
Rx: Power On	0.7 W
Rx: Standby Power	0.7 W
Tx: Power On	16 W
Downlink Data Rate	8000 (kbps)

Table 12: UC4 – EPS and Payload Parameters

EPS	
Battery Initial Fill	70 Amp-Hours
Battery Capacity	80 Amp-Hours
Solar Panel Area	0.2 m^2
Solar Panel Efficiency	0.17 (%/100)
Solar Panel Voltage	24 V
Depth of Discharge	50%
Overcharge	7%

Payload(s)	
Payload 1: Power On	40 W
P1: Operating Data Rate	20 kilobytes/sample time
P1: Standby Power	15 W
P1: Standby Data Rate	0 bytes/sample time
Payload 2: Power On	0 W
P2: Operating Data Rate	0 bytes/sample time
P2: Standby Power	0 W
P2: Standby Data Rate	0 bytes/sample time

Table 13: UC4 – ADCS and Structures Parameters

ADCS	
Magnetometer: Power On	1 W
Star Tracker: Power On	1 W
Sun Sensor: Power On	1 W
Torque Coil X: Power On	2 W
Torque Coil Y: Power On	2 W
Torque Coil Z: Power On	2 W
Reaction Wheels: Max Power	5.5 W
RW: Zero-torque Power	1.12 W
RW: Max Momentum	0.015 N-m-s
RW: Moment of Inertia	2.1e-5 kg-m^2
RW: Initial RPM	0 rpm

Structures	
Ixx MOI	0.188 kg-m^2
Ixy MOI	0 kg-m^2
Ixz MOI	0 kg-m^2
Iyy	0.183 kg-m^2
Iyz	0 kg-m^2
Izz	0.009968 kg-m^2
Center of Mass X	0.122 m
Center of Mass Y	0.110 m
Center of Mass Z	0.183 m
Total Mass	10.9 kg

3.7.5 Power Positive in Survival: Use-Case #5 (UC5)

In this case, a spacecraft ends up in Survival mode, but the CONOPS allows for it to recover to SunSafe if it meets certain criteria. This case is useful to determine if the spacecraft is power positive in tumble; a key performance parameter of any spacecraft design. Several modifications are needed to the state diagram to accomplish the desired behavioral response, as can be seen in Fig. 15.

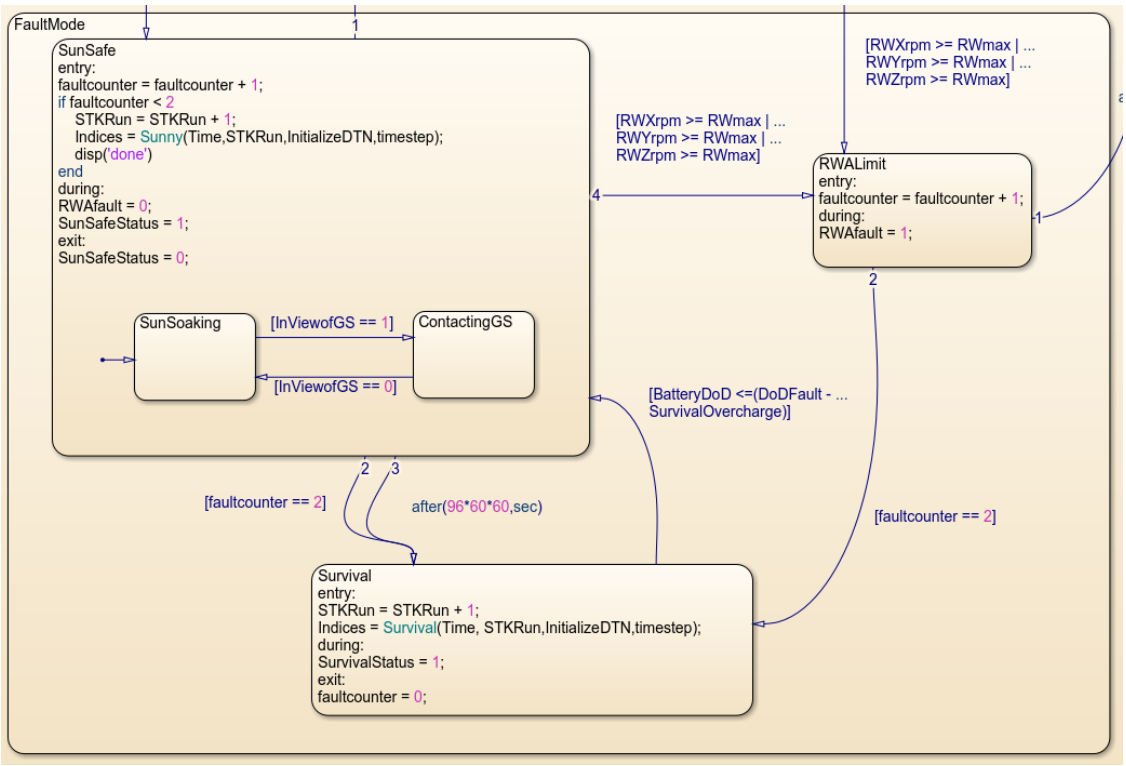

Figure 15: UC5 Fault Mode State Diagram, DyLoMMT
(Stateflow screenshot, with additions)

First, the Survival state is modified to allow for autonomous spacecraft recovery from Survival mode. This is designed to be an overcharge-based trigger, like the one that allows recovery from SunSafe. However, a new value is added to allow the modeler to set a different overcharge requirement to recover from Survival, as since Survival is a more serious fault mode, extra recovery of the batteries should be allowed for. This recovery from Survival does not return straight to Mission mode, but instead back to SunSafe, from which the spacecraft can be in contact with its ground station(s) and then, if the recovery trigger is met, the spacecraft will continue to recover to Mission mode.

Next, the criteria which thrusts the spacecraft into Survival is modified. The *faultcounter* value, which previously kept a running tally of fault modes entered, is now

reset each time the spacecraft enters Mission mode or exits Survival. This allows the *faultcounter* to serve as an indication of multiple fault states existing simultaneously (SunSafe and RWALimit in this case). When the spacecraft is already in the SunSafe fault mode, should the reaction wheels become saturated, the RWALimit fault mode will be entered, only now the *faultcounter* value will not have been reset, triggering the spacecraft to enter Survival. In Survival, the spacecraft cuts power to everything but the CD&H and Receiver systems, which is reflected in its *environmental model* which sets the spacecraft spinning about each body axis at 2 degrees per second to represent tumbling.

The final changes made are those allowing the spacecraft to enter the RWALimit mode from either Mission mode or SunSafe. Again, if entering from SunSafe, the spacecraft will then proceed into Survival and will only recover once/if the recovery conditions are met. To engage the extra cycling of the RAWLimit mode, the reaction wheels' moment of inertia is reduced by approximately 20%.

Table 14: UC5 – CD&H and TT&C Parameters

CD&H	
Bus SoH Data Rate	10 bytes/sample time
Payload SoH Data Rate	10 bytes/sample time
Extra Data Rate	0 bytes/sample time
Memory Full Status	10 MB
CDH On Power	3.5 W
Memory Initial Status	0 bytes

TT&C	
Rx: Power On	0.7 W
Rx: Standby Power	0.7 W
Tx: Power On	16 W
Downlink Data Rate	8000 (kbps)

Table 15: UC5 – EPS and Payload Parameters

EPS	
Battery Initial Fill	70 Amp-Hours
Battery Capacity	80 Amp-Hours
Solar Panel Area	0.2 m^2
Solar Panel Efficiency	0.17 (%/100)
Solar Panel Voltage	24 V
Depth of Discharge	50%
Overcharge	15%
Survival Overcharge	20%

Payload(s)	
Payload 1: Power On	40 W
P1: Operating Data Rate	20 kilobytes/sample time
P1: Standby Power	15 W
P1: Standby Data Rate	0 bytes/sample time
Payload 2: Power On	0 W
P2: Operating Data Rate	0 bytes/sample time
P2: Standby Power	0 W
P2: Standby Data Rate	0 bytes/sample time

Table 16: UC5 – ADCS and Structures Parameters

ADCS	
Magnetometer: Power On	1 W
Star Tracker: Power On	1 W
Sun Sensor: Power On	1 W
Torque Coil X: Power On	2 W
Torque Coil Y: Power On	2 W
Torque Coil Z: Power On	2 W
Reaction Wheels: Max Power	5.5 W
RW: Zero-torque Power	1.12 W
RW: Max Momentum	0.015 N-m-s
RW: Moment of Inertia	1.7e-5 kg-m^2
RW: Initial RPM	0 rpm

Structures	
Ixx MOI	0.188 kg-m^2
Ixy MOI	0 kg-m^2
Ixz MOI	0 kg-m^2
Iyy	0.183 kg-m^2
Iyz	0 kg-m^2
Izz	0.009968 kg-m^2
Center of Mass X	0.122 m
Center of Mass Y	0.110 m
Center of Mass Z	0.183 m
Total Mass	10.9 kg

3.7.6 Mission with Thruster Burn: Use-Case #6 (UC6)

This final use-case leverages the adjustments to the state machine developed for UC5 with the initial parameters used in UC1 to evaluate a mission-capable spacecraft design over a mission that now involves a thruster burn five days into the simulation. This is intended to initiate a repropagation based on not a mode change, but a maneuver completed by the spacecraft. To accomplish this new feature, the state machine was again modified to include a time-triggered state called Thrusting (Fig. 16) to generate the necessary repropagation script. To allow for easier visualization of the change to the data,

the change in velocity imparted via the burn is set to 150 m/s along each principle direction, or a combined thrust magnitude of 260 m/s, which is a little large for a spacecraft of this size.

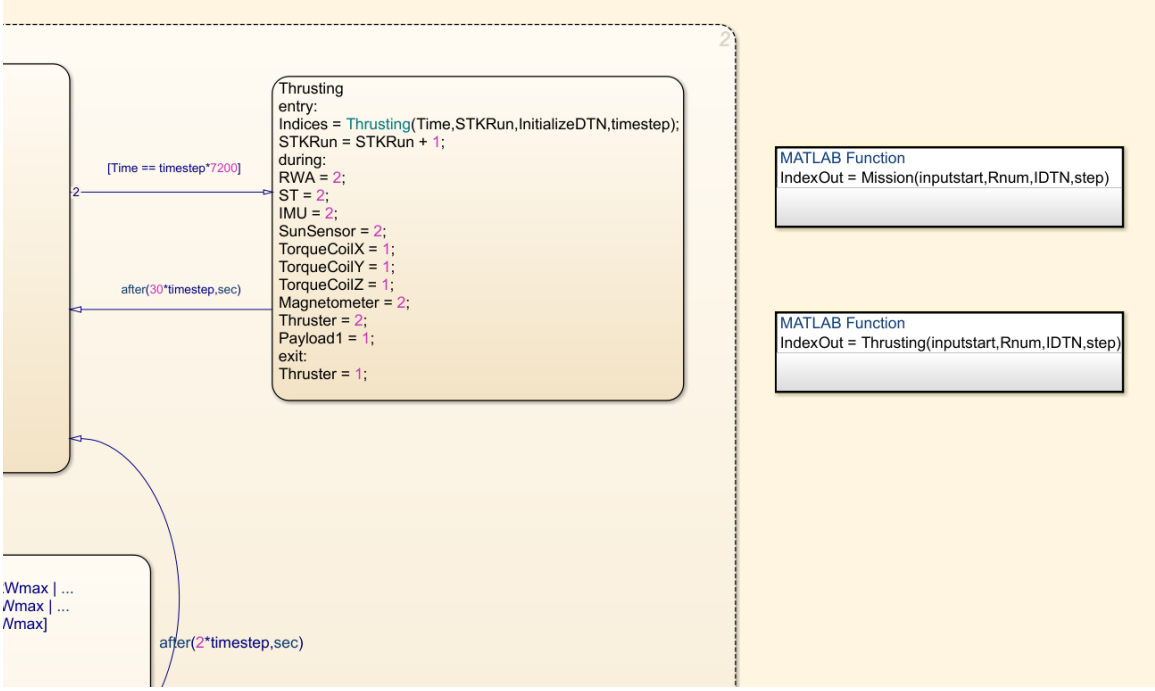

Figure 16: UC6 State Diagram, DyLoMMT (Stateflow screenshot)

Table 17: UC6 – CD&H and TT&C Parameters

CD&H	
Bus SoH Data Rate	10 bytes/sample time
Payload SoH Data Rate	10 bytes/sample time
Extra Data Rate	0 bytes/sample time
Memory Full Status	10 MB
CDH On Power	3.5 W
Memory Initial Status	0 bytes

TT&C	
Rx: Power On	0.7 W
Rx: Standby Power	0.7 W
Tx: Power On	16 W
Downlink Data Rate	8000 (kbps)

Table 18: UC6 – EPS and Payload Parameters

EPS	
Battery Initial Fill	70 Amp-Hours
Battery Capacity	80 Amp-Hours
Solar Panel Area	0.2 m^2
Solar Panel Efficiency	0.23 (%/100)
Solar Panel Voltage	24 V
Depth of Discharge	50%
Overcharge	15%
Survival Overcharge	20%

Payload(s)	
Payload 1: Power On	27 W
P1: Operating Data Rate	20 kilobytes/sample time
P1: Standby Power	9 W
P1: Standby Data Rate	0 bytes/sample time
Payload 2: Power On	0 W
P2: Operating Data Rate	0 bytes/sample time
P2: Standby Power	0 W
P2: Standby Data Rate	0 bytes/sample time

Table 19: UC6 – ADCS and Structures Parameters

ADCS	
Magnetometer: Power On	1 W
Star Tracker: Power On	1 W
Sun Sensor: Power On	1 W
Torque Coil X: Power On	2 W
Torque Coil Y: Power On	2 W
Torque Coil Z: Power On	2 W
Reaction Wheels: Max Power	5.5 W
RW: Zero-torque Power	1.12 W
RW: Max Momentum	0.015 N-m-s
RW: Moment of Inertia	2.1e-5 kg-m^2
RW: Initial RPM	0 rpm
ThrusterON Power	20 W

Structures	
Ixx MOI	0.188 kg-m^2
Ixy MOI	0 kg-m^2
Ixz MOI	0 kg-m^2
Iyy	0.183 kg-m^2
Iyz	0 kg-m^2
Izz	0.009968 kg-m^2
Center of Mass X	0.122 m
Center of Mass Y	0.110 m
Center of Mass Z	0.183 m
Total Mass	10.9 kg

3.8 Summary

This chapter described how this research intends to meet the objective of identifying when a dynamic modeling approach should be chosen over a static approach. The start of the chapter discussed the working details of the two main software programs utilized by the LMMT (and therefore the DyLoMMT), followed by a brief overview of the changes developed for the DyLoMMT. Finally, it described a mission representative reference scenario chosen for this research and diagrammed the six use-cases developed

to evaluate the software with. Section 3.7 (which covered the use-cases chosen) provides the structure by which Chapter 4: Analysis and Results is constructed. This list of use-cases is not an exhaustive one, and several opportunities have arisen for future work in this area. More details on applicable future work can be found in Sections 5.3 and 5.4.

IV Analysis and Results

4.1 Chapter Overview

This chapter details the results of the use-cases introduced in Chapter 3. Each DyLoMMT simulation generates a full set of telemetry, but only certain values are selected for display in figures. This was how the software preceding the DyLoMMT output its figures via the *makefigures.m* file, and this file remains generally unchanged (some formatting was altered to dock the figures instead of displaying them in a free-floating manner, as it originally coded). Each section will discuss the respective use-case's results and then present the figures. It may be of note to the reader that each use-case went through a series of iterations until arriving at the parameters and state diagrams reported in Chapter 3. Therefore, while there are no 'failed' use-cases reported in this chapter, this does not imply that these parameters/state diagrams were simply an unbroken string of good guesses which happened to achieve their respective behavioral goals, or that the software somehow guides the modeler into designing the correct mission. Varying degrees of effort were required to construct and then tune each use-case.

4.2 Generalized Imaging Mission: UC1

This use-case is a baseline whose results should be similar to something produced with the LMMT, as no repropagation runs should be initiated. As can be seen in Fig. 17, the only Fault Mode triggered was a brief RWALimit status as the reaction wheels became saturated late in the simulation (~6.8e5 epoch seconds). The remaining figures show exactly what one would expect for a fully mission-capable design: cyclic power

generation from the solar arrays, cyclic ground station downlink rates, cyclic imaging data rates as the spacecraft passes over targets, cyclical reaction wheel rates, and good charge on the batteries throughout the length of the simulation. Figures 17 – 20 make up the full standard output produced by the LMMT's *makefigures.m* file. In subsequent use-cases in this thesis, only the relevant figures are included.

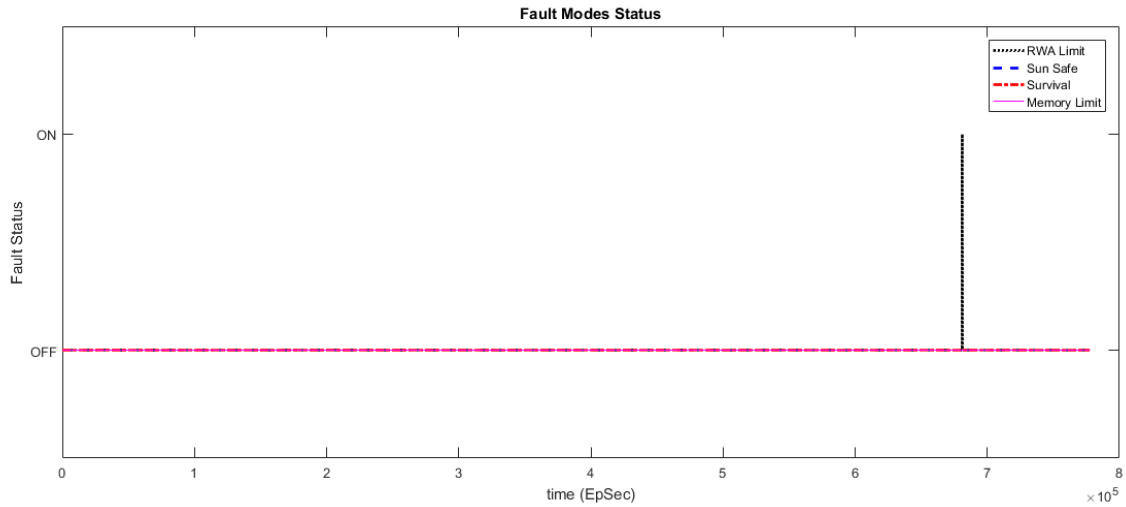

Figure 17: Fault Modes Triggered – UC1, DyLoMMT

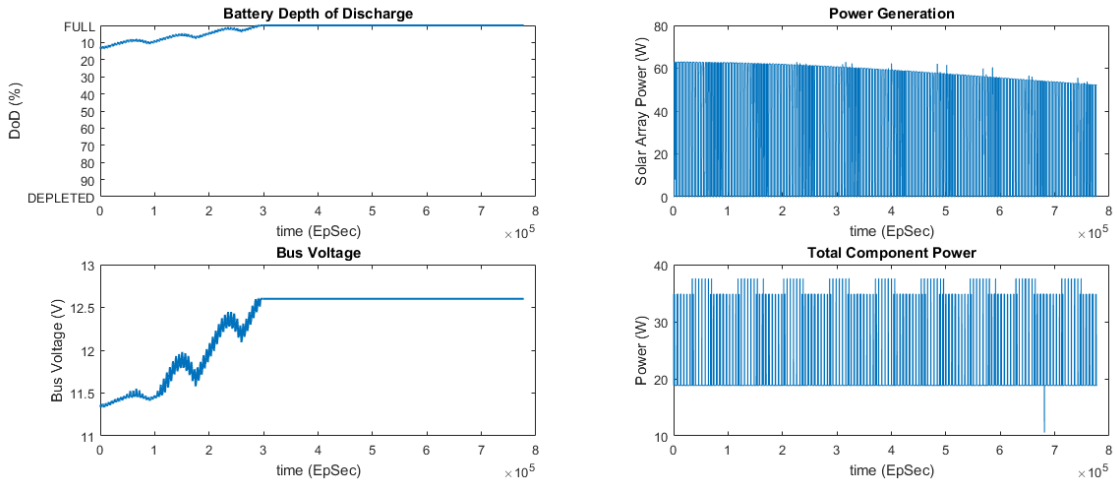

Figure 18: EPS Telemetry – UC1, DyLoMMT

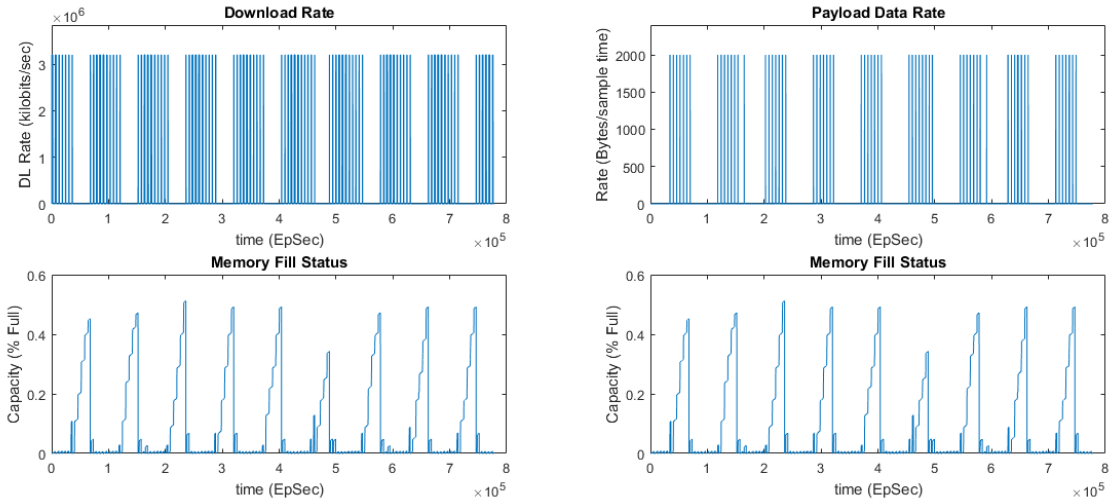

Figure 19: Data Usage – UC1, DyLoMMT

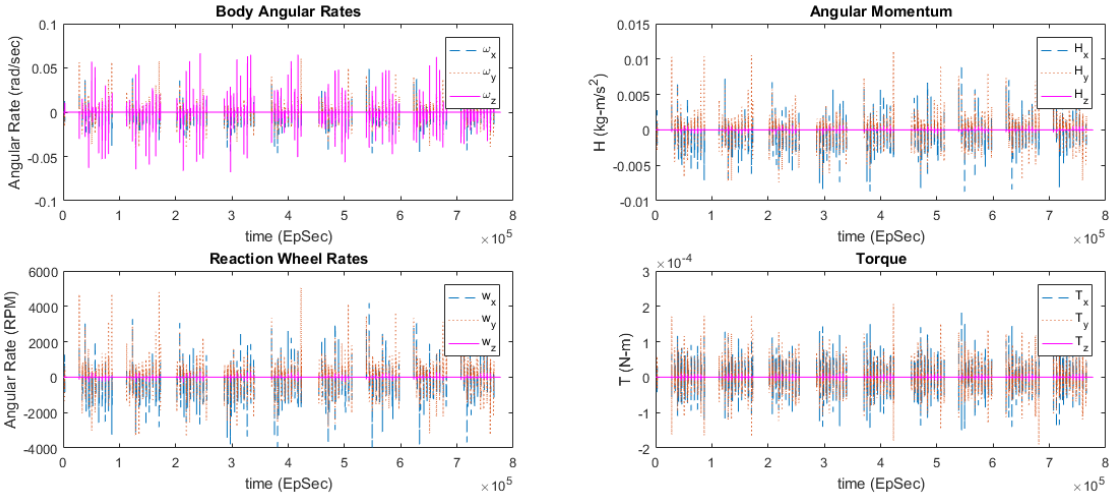

Figure 20: ADACS Readings – UC1, DyLoMMT

The static variant of this use-case was also run, and a comparison plot of the X vector magnitude is provided in Fig. 21. The two sets of data overlap each other perfectly, indicating that a dynamic approach to this use-case would not provide extra fidelity. This is to be expected, as there were no repropagations. There are many other factors to be compared between the static and dynamic approaches *environmental model* data, but they are left out from this thesis.

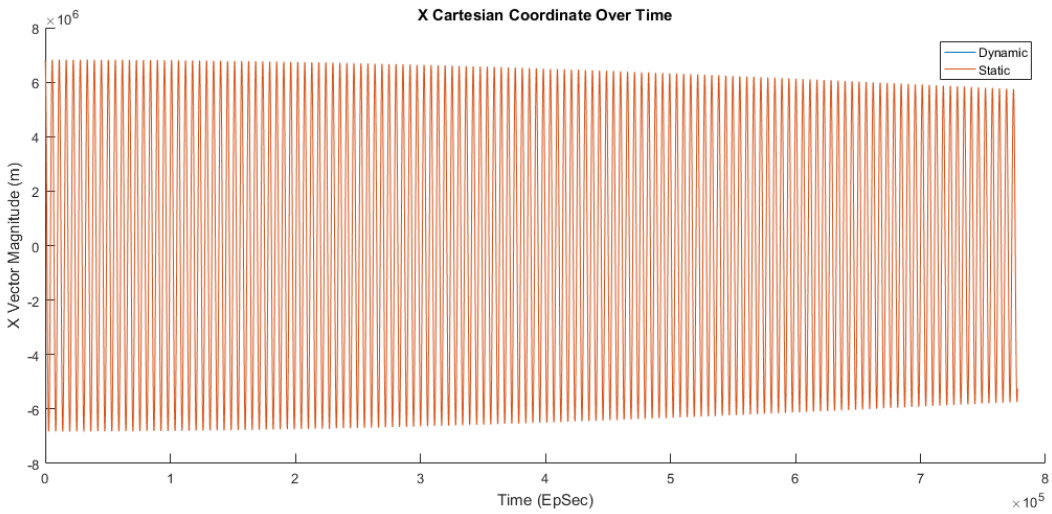

Figure 21: Static vs Dynamic Plot, X Magnitude – UC1, DyLoMMT

4.3 Straight into Fault: UC2

As this use-case was to demonstrate the spacecraft's mode state moving downward into the fault modes, a new payload power requirement was levied on the design (40 Watts while operating, 15 Watts while in standby), and the solar panel efficiency was dropped to a much lower/more conservative value (11%). The latter could represent cheaper solar cells, faults in manufacturing, on-orbit degradation, or perhaps some combination of the set. With these new parameters input to the model, Fig. 22 reports the spacecraft's mode state moving from normal ops (no Fault Status) into SunSafe (~1.0e5 epoch seconds) and then into Survival mode (~4.4e5 epoch seconds). It is important to remember that the spacecraft cannot exist in two of these states simultaneously, though Fig. 22 may appear to suggest otherwise. Both the transition into SunSafe as well as that into Survival triggered repropagations as per the state machine logic.

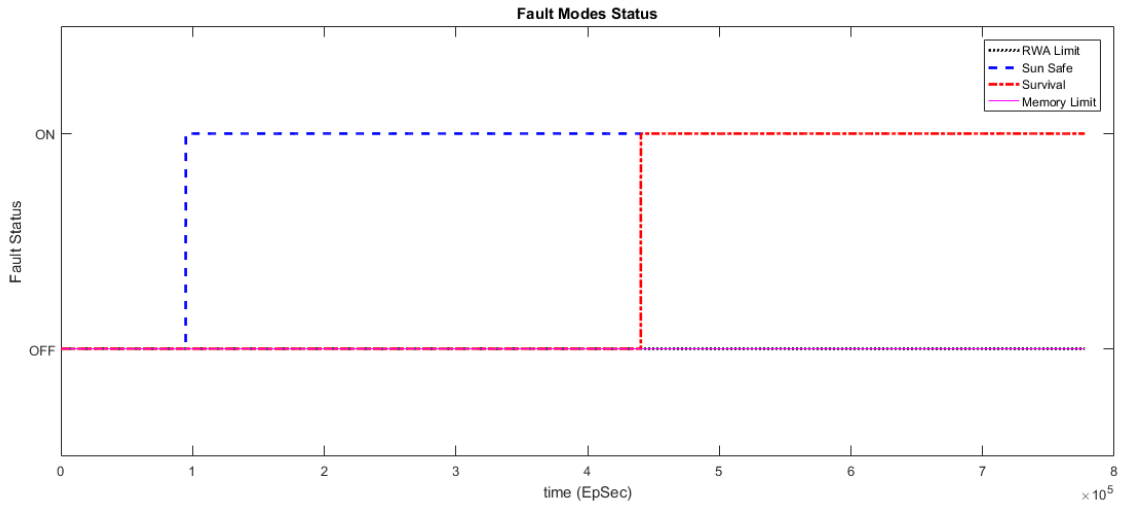

Figure 22: Fault Modes Triggered – UC2, DyLoMMT

Figure 23 clearly shows the effects of the mode changes on the rate of the Battery Depth of Discharge. At ~1.0e5 seconds, the rate of discharge noticeably decreases, and at ~4.4e5 seconds the batteries begin to recharge in Survival. This syncs with the Total Component Power graph detailing the power being used by the spacecraft.

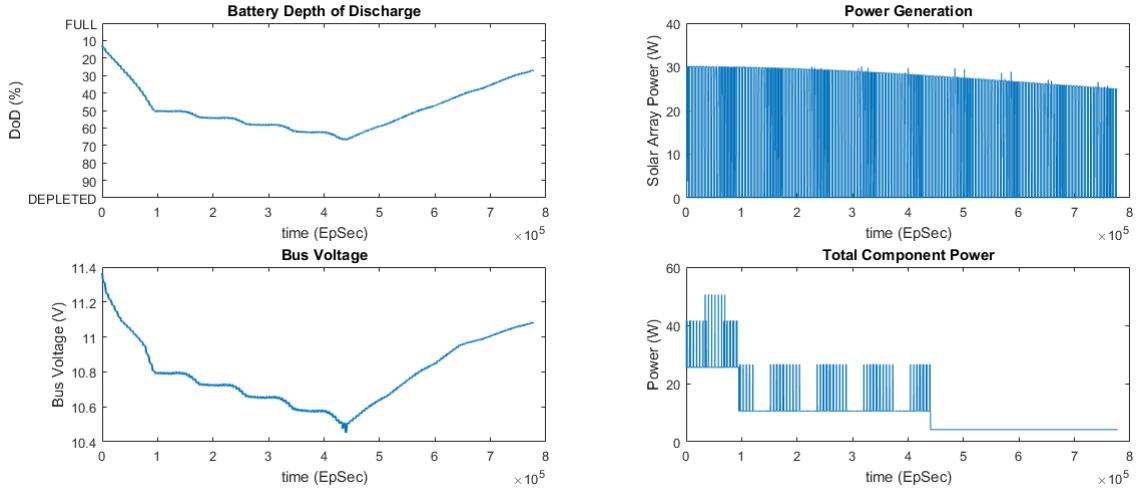

Figure 23: EPS Telemetry – UC2, DyLoMMT

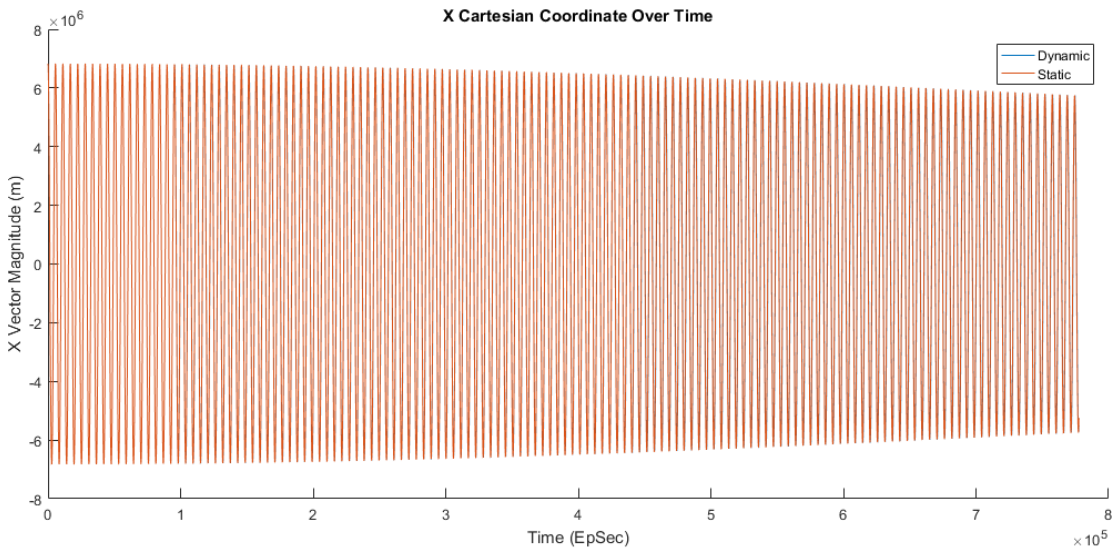

Figure 24: Static vs Dynamic Plot, X Magnitude – UC2, DyLoMMT

While the data in Fig. 24 overlaps perfectly as in UC1, there were repropagations run during this use-case's simulation, and so another *environmental model* data metric is checked for differences. This metric is the q1 quaternion of the spacecraft's attitude throughout the simulation. Figure 25 clearly shows that after the first repropagation at ~1e5 seconds, the dynamic model's q1 differs completely from its static counterpart. It also clearly shows the different state's attitude requirements (e.g. SunSafe: sun-pointing, Survival: tumble, etc.) as the spacecraft cycles through them. Therefore, if attitude is of interest to the modeler, a dynamic approach must be used in this case.

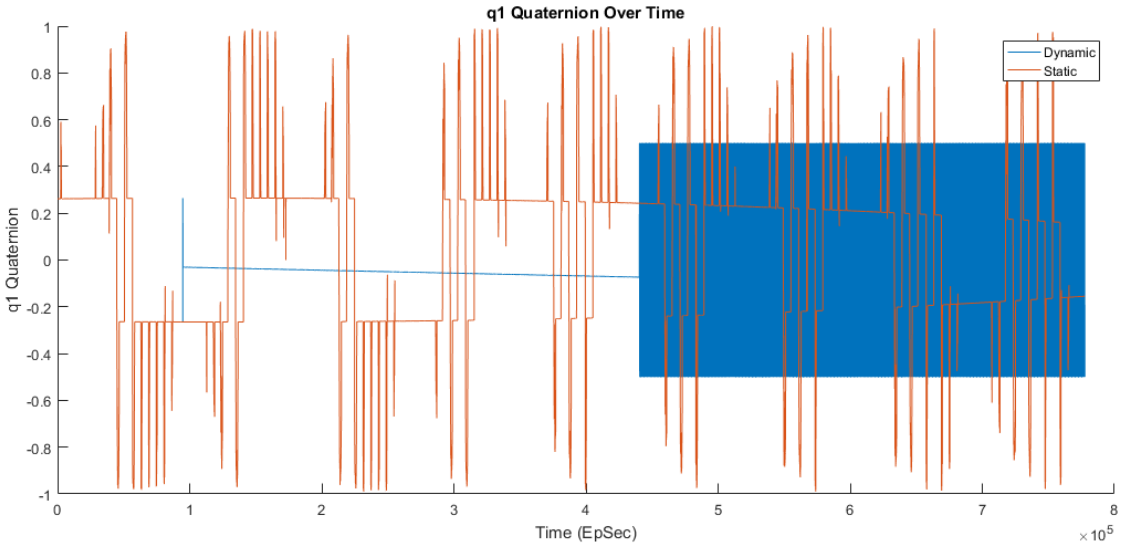

Figure 25: Static vs Dynamic Plot, q1 Quaternion – UC2, DyLoMMT

4.4 Into and Out of SunSafe, Then Survival: UC3

The purpose of this use-case is to demonstrate the capability of the DyLoMMT to cycle between normal operation and fault modes. The Survival mode has still not been modified to allow for autonomous recovery of the spacecraft, so the simulation moves between the SunSafe and Mission modes until Survival is triggered, which the spacecraft then remains in until the end of the simulation, as depicted in Fig. 26.

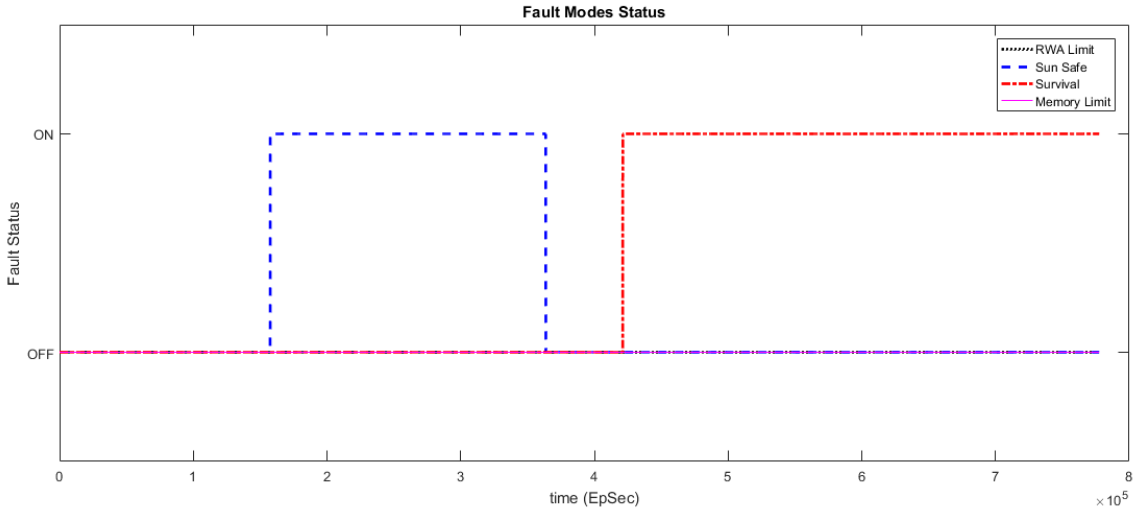

Figure 26: Fault Modes Triggered – UC3, DyLoMMT

Figure 27 details how the batteries sink to below the chosen DoD cutoff (50%), recharge in the SunSafe mode, and then dip back below the cutoff upon resumption of Mission mode power draw. Of note is the spacecraft's total power consumption as shown by the Total Component Power graph in Fig. 27; it is clear to see that, even in Survival mode, the spacecraft does not shut down all subsystems, instead attempting to keep the CD&H and receiving capability of the TT&C alive while there is power.

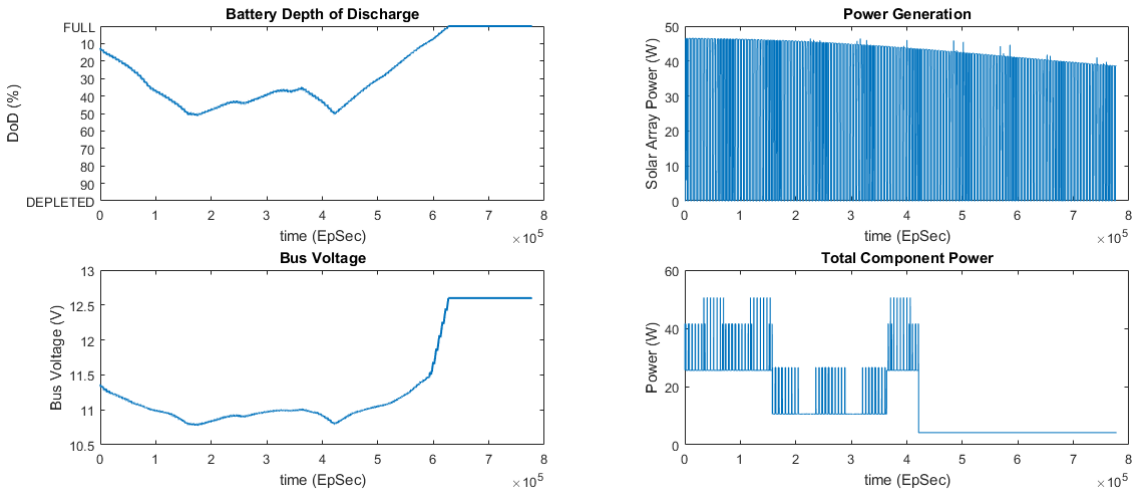

Figure 27: EPS Telemetry – UC3, DyLoMMT

Figure 28 shows how the payload data rate, the download rate, and the memory fill are all effected by the different modes. The memory storage needed climbs during orbits where no contact is made with the ground stations and decreases rapidly as soon as communication with the ground is reestablished.

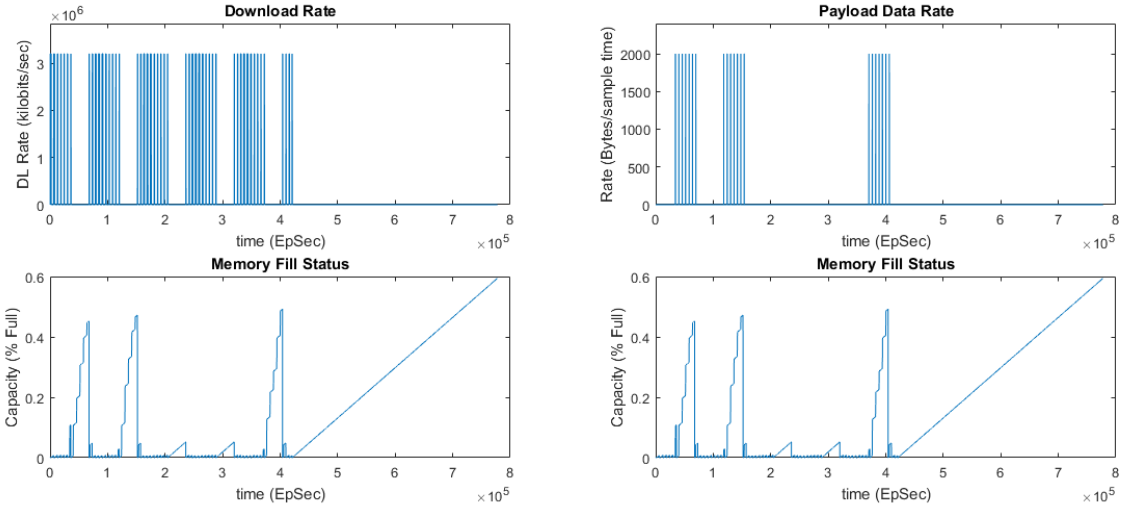

Figure 28: Data Usage – UC3, DyLoMMT

Figure 29 shows that once again the spacecraft's orbit remains virtually unchanged, with the only variation being a single time step lag introduced into the data by means of the time step used to initiate the repropagation. However, as in UC2, if the attitude of the spacecraft is of interest to the modeler, the static and dynamic approaches provide very different results (Fig. 30), and the dynamic approach may be more useful. If attitude is of no consequence to the modeler, then the repropagation triggers should reflect this and only be called if a spacecraft's action will have an impact on other *environmental model* metrics.

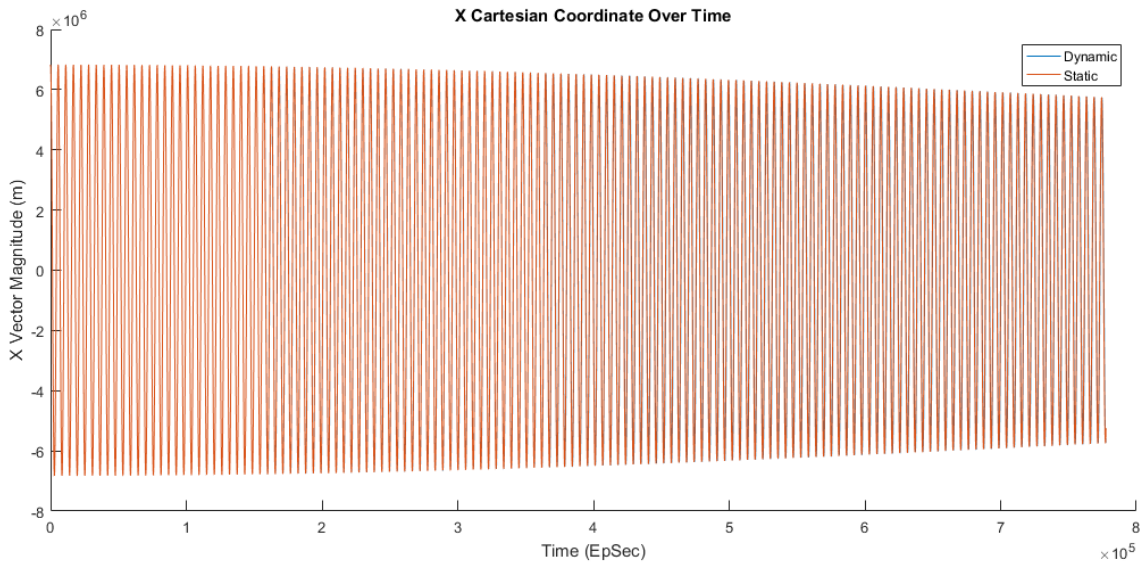

Figure 29: Static vs Dynamic Plot, X Magnitude – UC3, DyLoMMT

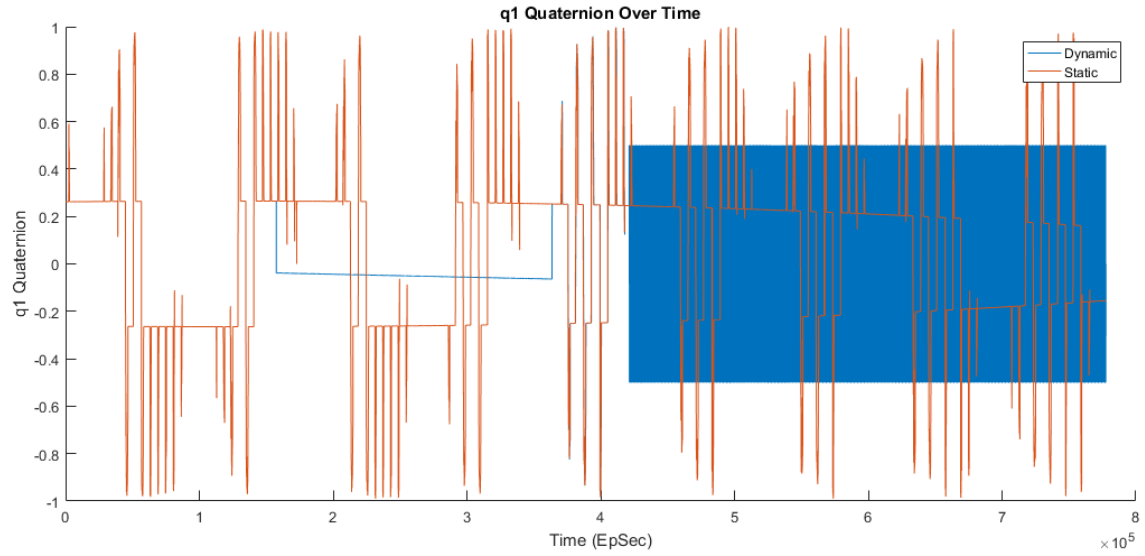

Figure 30: Static vs Dynamic Plot, q1 Quaternion – UC3, DyLoMMT

4.5 Into and Out of SunSafe Multiple Times: UC4

This next use-case demonstrates the ability to program the spacecraft CONOPS such that SunSafe can be entered as many times as triggered, with Survival only being triggered if other issues arise. This allows the repeated generation of Mission and

SunSafe *environmental model* repropagations that are spliced into the main data files as the simulation continues. UC4 clearly shows that the DyLoMMT can handle a repeated set of mode switches. The results are consistent with a spacecraft which has undersized solar arrays – when it operates in Mission mode, it drains its batteries, but upon assumption of SunSafe orientation and functionality, it recharges them and returns to Mission mode. However, it is easy to see in Fig. 31 that the spacecraft spends considerably more time in the SunSafe mode than it does accomplishing its mission, which suggests that while the spacecraft bus operations require less electrical power than the payload, they still use a large amount of the power generated via the solar arrays, causing a slow recharge rate. Or, this may be a limitation of the battery packs themselves, as the DyLoMMT will only allow them to charge at the rate specified by the modeler (in this case, 5.2 Amps per time step). This sort of analysis is what the DyLoMMT is well-suited to assist with.

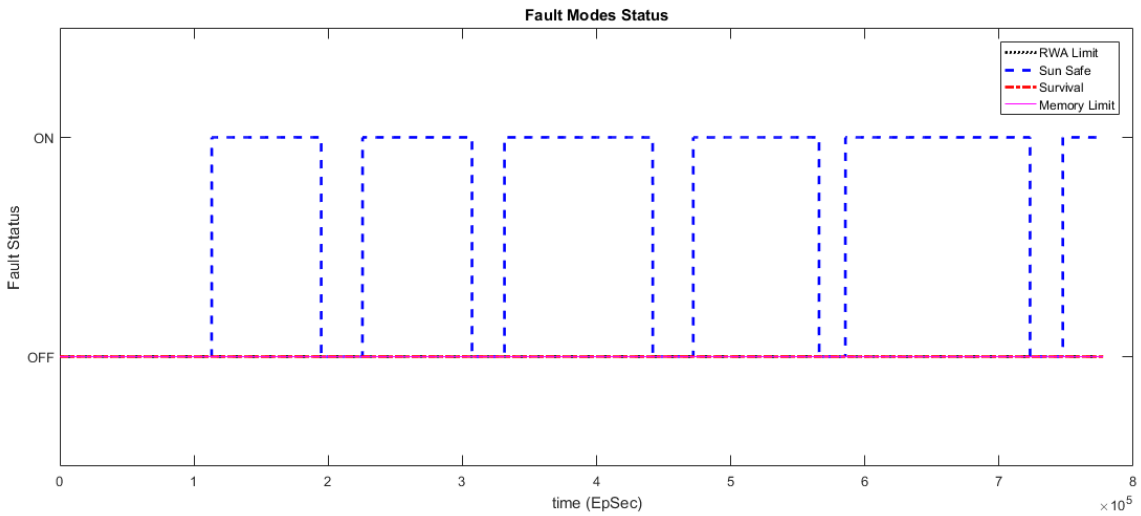

Figure 31: Fault Modes Triggered – UC4, DyLoMMT

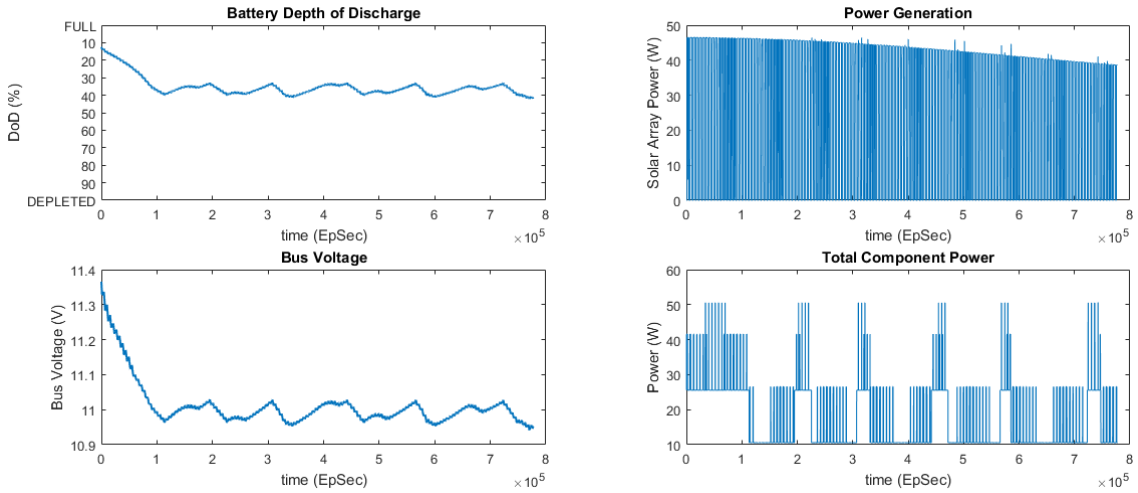

Figure 32: EPS Telemetry – UC4, DyLoMMT

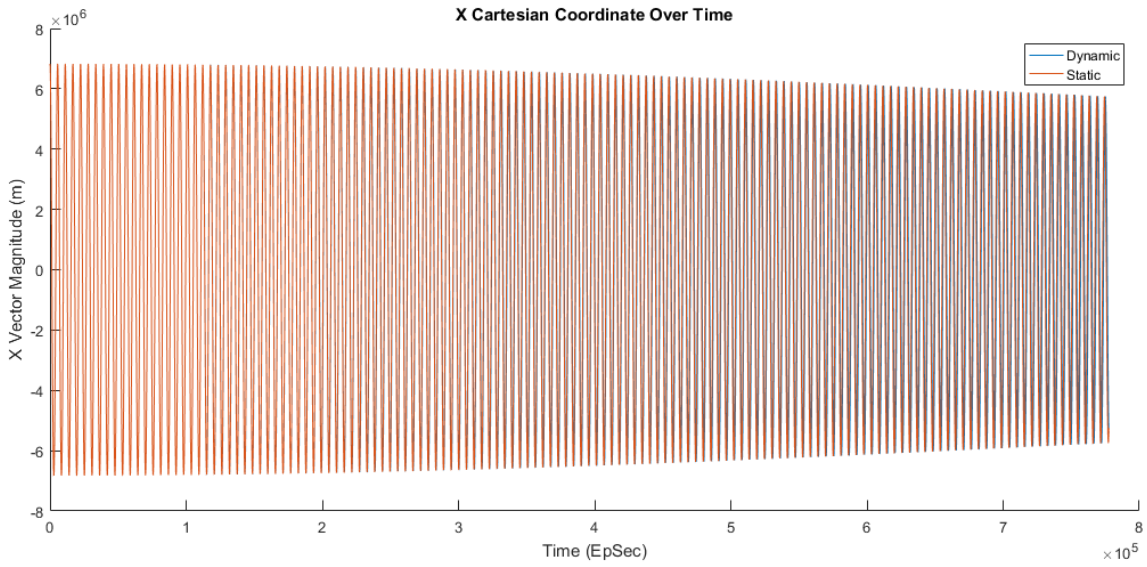

Figure 33: Static vs Dynamic Plot, X Magnitude – UC4, DyLoMMT

Figures 33 and 34 show the same trend as in the preceding use-case, the physical orbit of the spacecraft does not change, even with repropagations, but the spacecraft's attitude very much does.

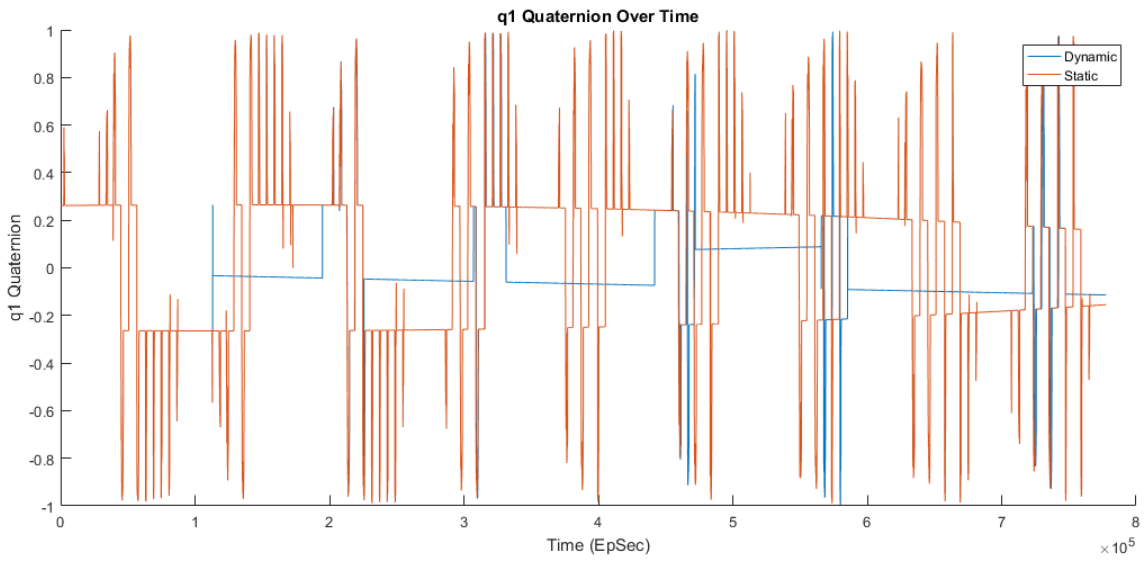

Figure 34: Static vs Dynamic Plot, q1 Quaternion – UC4, DyLoMMT

4.6 Power Positive in Survival: UC5

This next use-case demonstrates what is the most complicated state machine diagram designed for the DyLoMMT yet, as well as the capacity for the spacecraft to autonomously switch from Mission, SunSafe, RWALimit, and Survival modes. The state machine operates with the logic that the spacecraft would not be able to maintain SunSafe attitude if its reaction wheels were saturated.

Many of the improvements needed to the state machine of the DyLoMMT were discovered while iterating this use-case. For example, this use-case's set of iterations are where the issues with the payload configuration state loop were identified and fixed. The 10 time steps delay on wakeup for the payload (designed with only initial startup in mind) led to the payload repeatedly returning to the Standby state after being powered down, even during a fault mode, draining the spacecraft's batteries and sending it rapidly into Survival mode with no chance of return due to the continuous heavy power draw.

For detail on the solution, compare the initial attempt shown in Fig. 10 with the fully-updated version in Fig. 35. Figures 36 and 37 show the cycling of the spacecraft's mode states as SunSafe is triggered, the wheels saturate (which triggers Survival), then the batteries are allowed to recharge to the point where the spacecraft climbs back up to Mission mode, only to again cycle back down through to Survival.

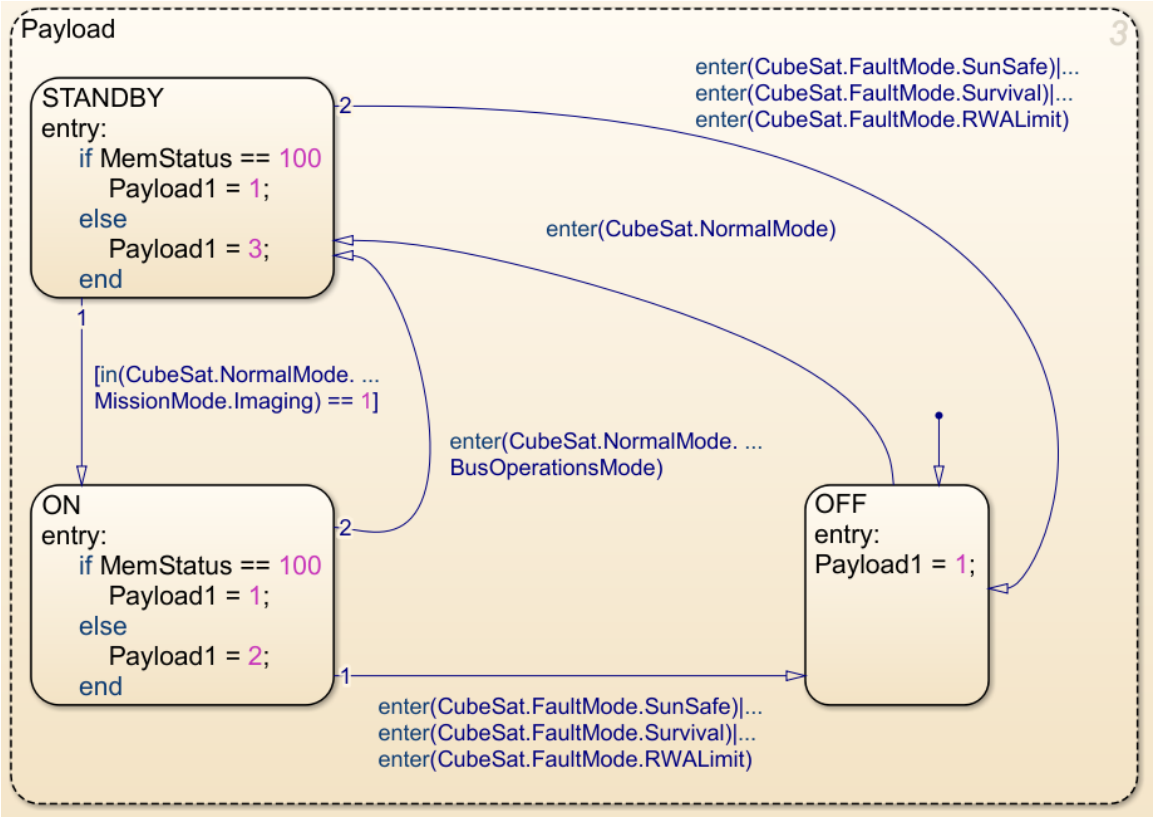

Figure 35: Payload Configuration State, DyLoMMT (Stateflow screenshot)

An interesting result is produced by UC5 – the spacecraft logically is triggered from SunSafe to Survival mode if its reaction wheels saturate, but once set into a sun-pointing attitude, very minor corrections should be needed from the wheels to keep this orientation, unless there is an outside torque present. There is no such torque coded into this use-case, so something else is causing the wheels to spin up and saturate. It appears

this is because the DyLoMMT's calculation side views the attitude passed in from the *environmental model* as a series of discrete positions to be moved between. So when faced with a constant motion orientation, the software still calculates that the wheels are required to continually update this orientation, when physically this is not the case.

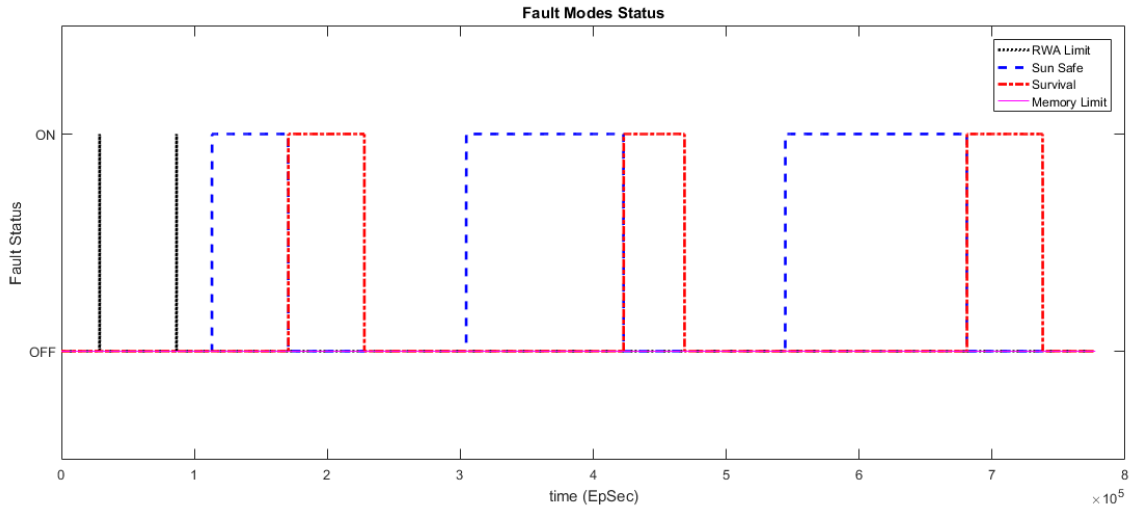

Figure 36: Fault Modes Triggered – UC5, DyLoMMT

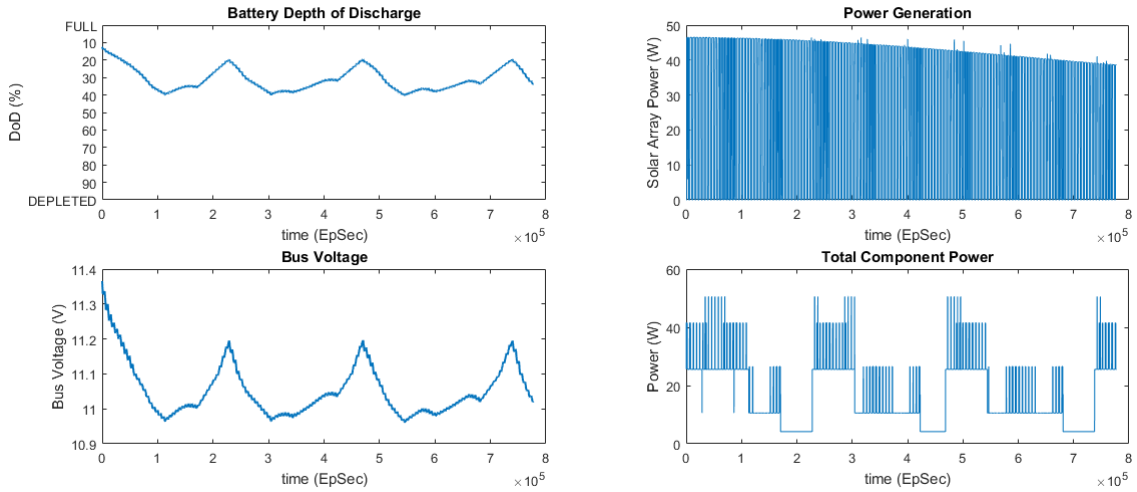

Figure 37: EPS Telemetry – UC5, DyLoMMT

As for a dynamic or static approach, the previous trend continues, with nothing perturbing the orbit of the spacecraft over the repropagations (Fig. 38) but dramatic

differences recorded in the spacecraft's attitude (Fig. 39). This use-case's q1 quaternion graph also provides a good example of the attitude requirements of each fault mode, with the SunSafe mode introducing a slow change in attitude, followed by the quick, cyclical changes seen in Survival mode as the spacecraft tumbles. This information is only seen in the Dynamic data of these use-cases.

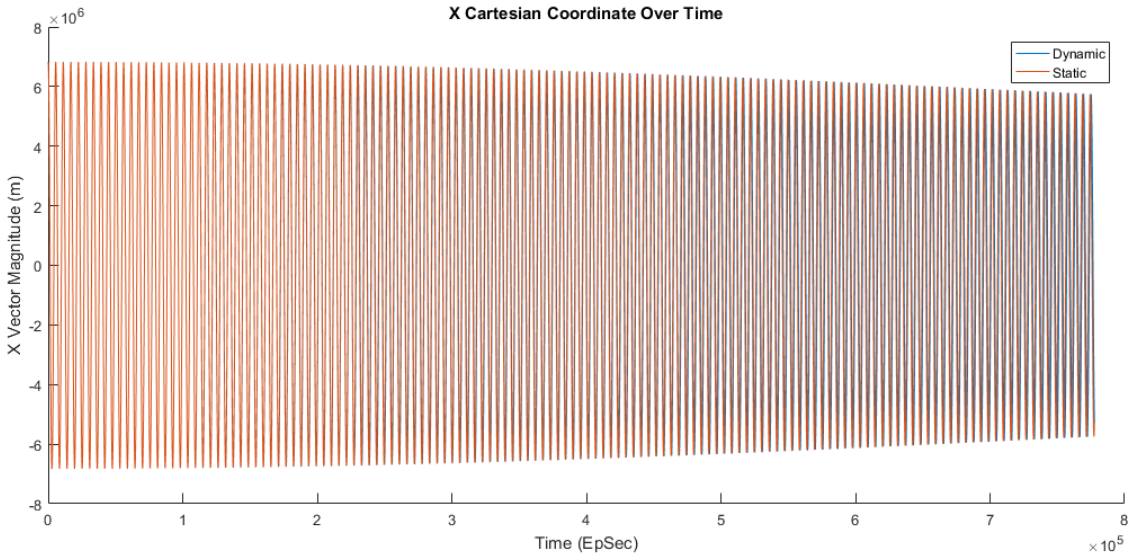

Figure 38: Static vs Dynamic Plot, X Magnitude – UC5, DyLoMMT

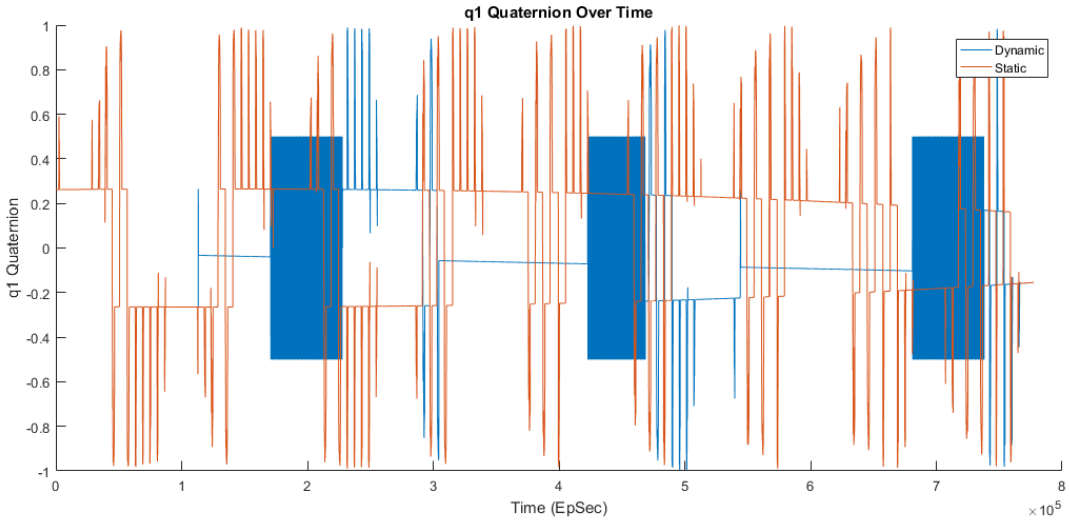

Figure 39: Static vs Dynamic Plot, q1 Quaternion – UC5, DyLoMMT

4.7 Mission with Thruster Burn: UC6

This final use-case introduces a repropagation not related to a fault mode, but instead to an action by the spacecraft, namely, firing a thruster five days into the mission. As Fig. 40 shows, only the occasional RWALimit fault was encountered, and therefore no mode change repropagations occurred. However, the day five thruster burn did occur and is seen in the power drop during the burn, when the payload was powered off and remained off until the next imaging opportunity (Fig. 41).

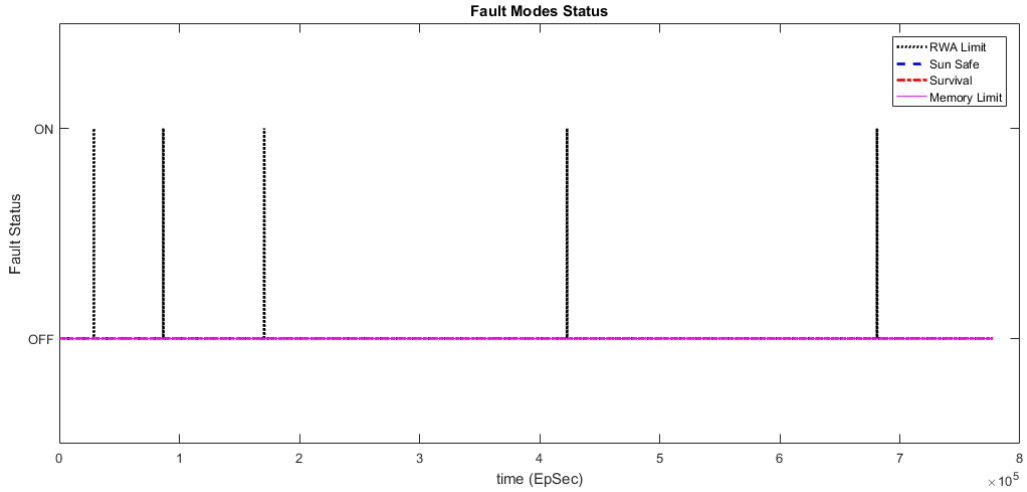

Figure 40: Fault Modes Triggered – UC6, DyLoMMT

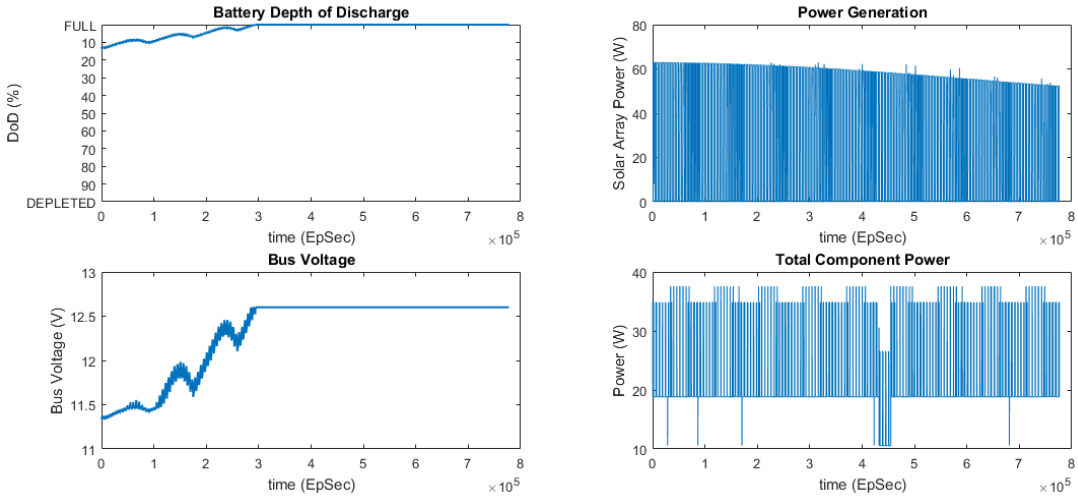

Figure 41: EPS Telemetry – UC6, DyLoMMT

This final case also demonstrates the first utility of the X coordinates metric. Clearly seen in Fig. 42 is the thruster burn's resulting change in the spacecraft's orbit. Again, while this change in velocity may be a little high for a spacecraft weighing in at just over 10kg, it serves to illustrate the utility of the repropagation. Of interest is the unexpected result of different attitudes between the static and dynamic simulations displayed in Fig. 43. This is understandable, as a new orbit would induce new pointing requirements for the various targets, but it is an interesting result as the repropagation did not induce an attitude change directly as in UC2–UC5, but it still appeared as an emergent response to the change in orbit.

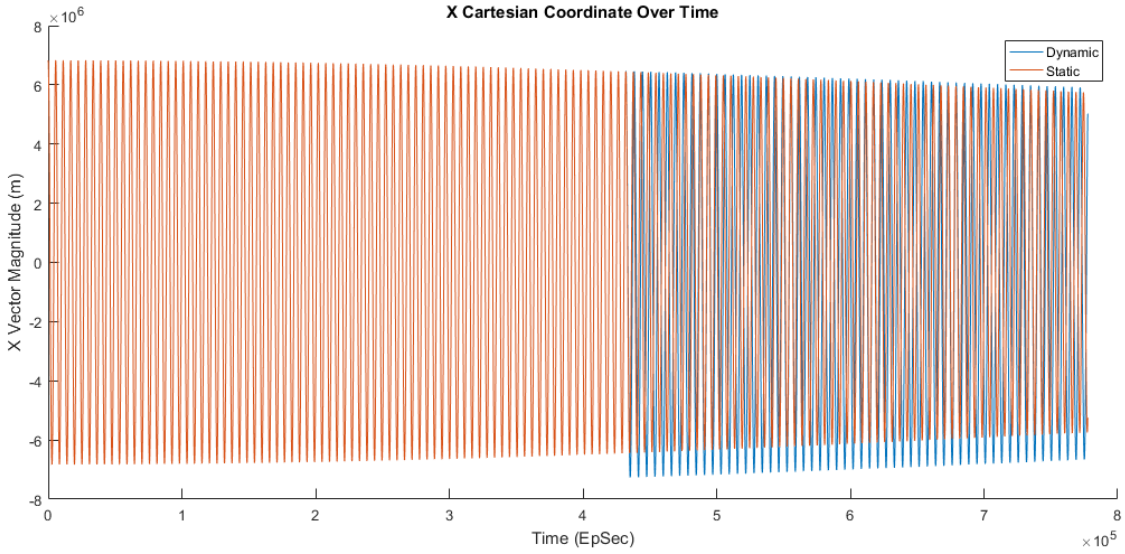

Figure 42: Static vs Dynamic Plot, X Magnitude – UC6, DyLoMMT

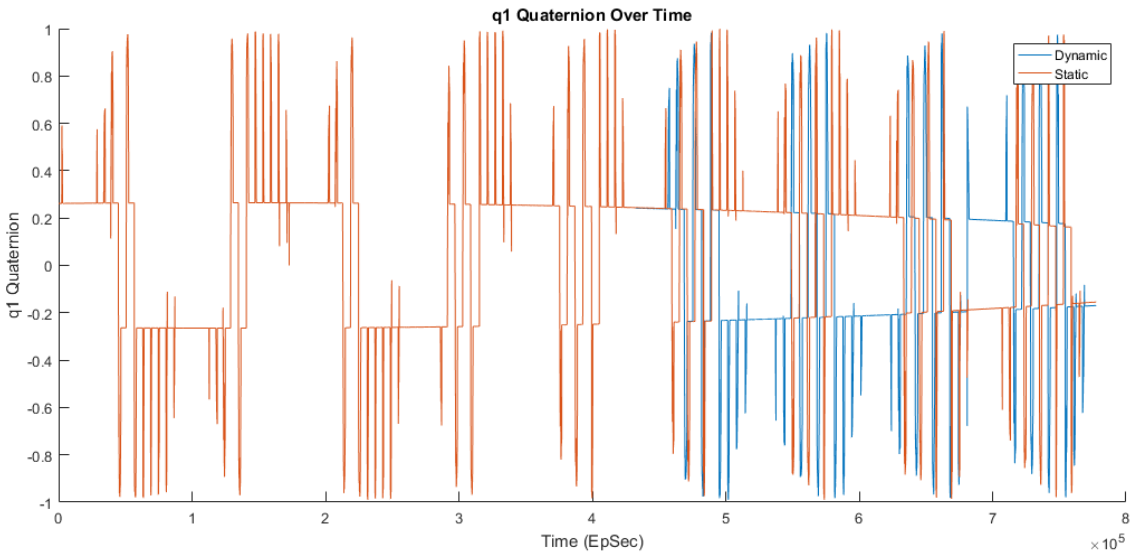

Figure 43: Static vs Dynamic Plot, q1 Quaternion – UC6, DyLoMMT

4.8 Research Objective Revisited/Impact of Results

This research sought to develop a method of identifying when a dynamic modeling approach would be more appropriate than a static modeling approach. The use-cases presented in this chapter illustrate that how a modeler defines the triggers for repropagation will often inform when a dynamic model should be utilized for added fidelity. For instance, if the modeler is only interested in coming up with a design that experiences no fault modes, then a simple static model will suffice, as the model will be iterated until no repropagation runs are exhibited and the data following a repropagation is of no interest. However, if it is the behaviors of the spacecraft in all modes of operation (some of which require repropagation runs) that is of interest to the modeler, then a dynamic approach will provide a more complete picture of how the spacecraft is interacting with its environment on-orbit. This also applies to simulations including spacecraft actions that may alter their spacecraft's interactions with the *environmental*

model data, such as the thruster burn in UC6. The DyLoMMT offers an improved CONOPS evaluation capability as compared to the static LMMT, as changes in mode hierarchy and triggers for modes are now reflected in updates to the *environmental model*.

With this new software, the model is capably of autonomously triggering repropagations of the *environmental model* using new mode conditions (as necessary) throughout the simulation. This in turn provides a closer match of spacecraft behavior to on-orbit conditions. In essence, the DyLoMMT adds the ability to model any *environmental model* consequences of actions taken by the spacecraft during the simulation. This research shows that mode-based CONOPS logic can drive repropagations in the DyLoMMT without limiting the decision-making autonomy of the state machine. The model (once constructed) can autonomously decide which mode it should exist in as well as decide if/when to repropagate if something has changed that makes the spacecraft's current perception of the *environmental model* data no longer current/relevant to the simulation. This capability should give modelers more confidence in the predictions of the spacecraft's behavior produced, particularly when simulating a model of a system which will trigger repropagations. If not intending to leverage repropagations, the generally quicker-running static model should be used.

4.9 Summary

This chapter provided the DyLoMMT-generated predictions of spacecraft performance and behaviors over the course of a 10-day mission-representative simulation. Full telemetry and figures were generated for each use-case, but only the

relevant were presented in this thesis, and changes to the state machine logic were discussed for each relevant use-case. A presentation of both static and dynamic variants of each use-case informed the chapter's conclusions.

V Conclusions and Recommendations

5.1 Chapter Overview

This final chapter speaks briefly to the significance of the research presented in this thesis and continues on to describe recommendations for future action to improve the software as well as future areas of research related to the DyLoMMT. The Recommendations for Action section covers ideas that are more concrete and not considered to be research opportunities in of themselves, whereas the Recommendations for Future Research section covers ideas which are more abstract and could serve to be their own research opportunities.

5.2 Conclusion and Significance of Research

The products of this research, the figures and data presented, are the result of an iterative effort (described in Chapter 3) which partially explains the lack of "failed" use-cases. With this research, either the concept was going to be feasible, in which case it would be implemented and produce results, or it was not. Once a general use-case could be simulated, it was only a matter of tuning the physical parameters of the model or the state machine logic to simulate other use-cases. The results presented in this thesis show that this research's objective was achieved, and the six use-cases serve to illustrate different behaviors that can be simulated with the finished modifications. As discussed in Section 4.8, the creation of the DyLoMMT allows dynamic spacecraft modeling which considers the consequences of the spacecraft's actions within the context of the environment it is operating in. This concept adds fidelity to and confidence in the predictions produced by the simulation. "All models are wrong, some are useful" is an

aphorism generally attributed to the statistician George E.P. Box [40] and gives some context to the significance of this research. The DyLoMMT does not a perfectly model a spacecraft's on-orbit behavior. No model will. So, in place of trying to convince the reader of that which is not, this research illustrates a feasible modeling tool which aims to be more useful by modeling the action-consequence pairing discussed above and discusses when this tool should be leveraged for higher-fidelity results. Moreover, this research shows that even with single-agent software such as the DyLoMMT, there is a wealth of information that can be ascertained from a dynamic modeling approach. This result should generalize to more complete dynamic modeling approaches such as AGI's AMOEBA plug-in. This is significant to any modeler wishing to model a mission whose spacecraft has modes which change how the spacecraft interacts with its environment or whose spacecraft regularly alters its perception of the environmental data.

5.3 Recommendations for Action

In Section 1.5.2 the scope and limitations of this research effort were set out, and a few provide good starting points for future action in the vein of this research that may not quite make future research foci in of themselves.

The calculations performed inside the DyLoMMT to produce the higher-level values which the state machine uses to operate have now undergone two iterations of students modifying and integrating them with other code. It may be prudent to audit the equations/blocks used and update them as necessary to ensure they accurately perform their functions. In particular, the several algebraic loops found in the ADACS subsystem should be given a closer look. These Simulink loops caused errors until augmented with a

unit step delay block, which circumvented Simulink throwing an error and quitting the simulation. Unfortunately, this also leaves a zero for the initial value, which is logged in their respective column in the telemetry file. This may or may not also delay each value one time step, which might concern some modelers. Finally, the cyclic nature of the power generated by the solar arrays suggests something in either the spacecraft's designed orbit in the *environmental model* or the method by which the attitude and sun vector are combined to produce a measure of the power generated have some bug that may be worth investigating. It may be that this is a seasonal or physically-valid effect, and the operation of the interaction between the *environmental model* and the state machine has been checked out and shown to be functioning as designed. This sort of software verification and validation is not research in of itself, but may be worth further action.

Further action also may be warranted to convert the DyLoMMT from what was designed and built as predominately research code into a more user-friendly tool. Although an effort has been made to responsibly document the scripting and coding done, there is much work that can be done to streamline the modeler's experience. The STK generation scripts are well-documented in terms of the purpose of each segment of code, but AGI's Connect has a syntax that is not intuitive nor exhaustively well-documented, which means there would be a learning curve should a modeler wish to alter the parameters of the scenarios in substantial ways (e.g. changing pointing vectors or adding components to the scenario that aren't currently modeled). The MATLAB code, in contrast, was directly modified from Loudermilk [6], and the documentation and modeler guidelines provided no longer always apply. Furthermore, given the nature of the glue-

coding techniques applied, not all code is efficiently written. A modeler who is more fluent in good coding practice may find themselves wondering why certain areas are coded as they are, and generally the reason is because it required the least amount of rewriting of previous, fully-functional code while still accomplishing the new purpose. The attempt was made to change only the pieces it was necessary to, and this no doubt has introduced inefficiencies in the software.

Another good example of work needed is how, in the DyLoMMT, STK opens and closes during each propagation run (initial or otherwise), presumably due to the interaction of the COM port connecting to STK and the function calls inside MATLAB. While each start-and-stop cycle costs mere seconds in terms of CPU time, this could become costly if long-duration simulations or those with significant amounts of repropagation runs are desired. It is possible that this delay is unavoidable, but this research has not collected enough information to make such a judgment. Another good example of cleaner coding is that (by virtue of converting the STK generation scripts into functions which receive a structure) the opportunity arises to convert much of what is currently hard-coded into each script into a structure which could be passed through from the *Master.m* file (such as the scenario's end time). This would allow the modeler the option to set these variables once, removing the need to carefully construct each repropagation script to ensure accuracy. Finally, adjusting the repropagation scripts to avoid the current time step slip in orbital data would give further clarity to an issue which will only get worse as more repropagations are added into the simulation.

5.4 Recommendations for Future Research

This research opens several opportunities for future research into the area of spacecraft modeling and simulation. The suggestions that follow are not an exhaustive list but should serve to give the reader some indication of ways to move forward from where this research left off.

One of the first and most directly-applicable efforts is to design more use-cases to simulate to test spacecraft designs and/or logic. This could relate to changing CONOPS to evaluate how a physical design reacts when different logic is presented to it, or it could relate to running the same CONOPS with differing physical designs against a common mission scenario to determine the ideal hardware parameters for the mission. One use-case in particular that should bear relevance to most spacecraft missions is a loss-of-comms event, and how the spacecraft would react to this. This use-case could be developed by altering the *environmental model* to reflect a temporary loss of ground station or by an inject into the state machine removing power to the subsystem in question. Either of these techniques (altering the *environmental model* and/or injects into the state machine) should prove useful to expanding the range of use-cases available to test. Manual injects into the software may be accomplished by introducing a series of breaks triggered at certain time or activity points in the state machine which call for operator input. The operator could then specify some sort of flag that would alter the model in some manner (as with the example of the loss-of-comms event) and then the simulation would proceed with the new configuration, reacting accordingly.

Another good area of research would be a sensitivity analysis of how much of a difference in predictions is made between the LMMT and the DyLoMMT. Are certain

styles/durations/complexities of missions more easily and as-accurately modeled using the less-complicated LMMT? How close is close enough for various metrics? Along with this line of reasoning comes the question: how much more fidelity should be added to the STK scenarios for optimal return on investment? Currently, there are slew-limiting factors built into the STK scenarios; are things of this nature worth the extra effort needed by the modeler to interpret and use correctly? What if the spacecraft has a directional antenna, how much of an effect does adding its slewing maneuvers into STK have on the reaction wheel assembly?

This development of this software provides many different avenues to approach, and research of this nature will continue to be of interest and value to the space community.

Appendix

A1. Scenario_Creation.m STK Generation Connect/MATLAB Code

```
function [outputSTK] = Scenario_Creation(inputSTK)
dbstop if error
```

Script for Creating and Populating New STK Scenario

```
app = actxserver('STK11.application');
root = app.Personality2;
%Create new scenario, second string part is scenario name
scenario = root.Children.New('eScenario','STKScenario');
% Maximize the window STK opens in and set Step Size for all scenario
reports and animations
root.ExecuteCommand('Application / Maximize');
step = inputSTK.step;
%Set the time period desired for the scenario, and reset graphics
scenario.SetTimePeriod(inputSTK.InitializeDT,'3 Oct 2016
12:00:00.000')
root.ExecuteCommand(sprintf('SetAnimation * TimeStep %d',step));
root.ExecuteCommand('Animate * Reset');
% Set correct units for ConnectReports
root.ExecuteCommand('Units_set * ConnectReportUnitsFlag on');
root.ExecuteCommand('Units_Set * ConnectReport Date "EpochSeconds"')
Not enough input arguments.
Error in Scenario_Creation (line 15)
step = inputSTK.step;
```

Create new facilities for GS contact

```
AFIT = scenario.Children.New('eFacility','AFIT');
SDL = scenario.Children.New('eFacility', 'SDL');
NPS = scenario.Children.New('eFacility', 'NPS');
UHi = scenario.Children.New('eFacility', 'UHawaii');
% Update the geodetic location so all facilities aren't at AGI HQ
%AFIT IAgFacility facility: Facility Object
AFIT.Position.AssignGeodetic(39.7819,-84.0822,0) % Latitude,
Longitude, Altitude
% Set altitude to height of terrain
AFIT.UseTerrain = true;
% Add sensor for coverage
AFIT_S = AFIT.Children.New('eSensor', 'Sensor1');
root.ExecuteCommand('Define */Facility/AFIT/Sensor/Sensor1 SimpleCone
80');
%SDL IAgFacility facility: Facility Object
SDL.Position.AssignGeodetic(41.7607,-111.819,0) % Latitude, Longitude,
Altitude
% Set altitude to height of terrain
SDL.UseTerrain = true;
% Add sensor for coverage
SDL_S = SDL.Children.New('eSensor', 'Sensor2');
root.ExecuteCommand('Define */Facility/SDL/Sensor/Sensor2 SimpleCone
80');
%NPS IAgFacility facility: Facility Object
NPS.Position.AssignGeodetic(36.5944,-121.875,0) % Latitude, Longitude,
Altitude
% Set altitude to height of terrain
NPS.UseTerrain = true;
% Add sensor for coverage
NPS_S = NPS.Children.New('eSensor', 'Sensor3');
root.ExecuteCommand('Define */Facility/NPS/Sensor/Sensor3 SimpleCone
80');
```

```
%UoHawaii IAgFacility facility: Facility Object
UHi.Position.AssignGeodetic(21.3161,-157.886,0) % Latitude, Longitude,
Altitude
% Set altitude to height of terrain
UHi.UseTerrain = true;
% Add sensor for coverage
UHi_S = UHi.Children.New('eSensor', 'Sensor4');
root.ExecuteCommand('Define */Facility/UHawaii/Sensor/Sensor4
SimpleCone 80');
```

Create a new satellite

```
LEOsat = scenario.Children.New('eSatellite', 'LEOsat');
%Generate command to set up orbital elements for satellite
%Following sat name: Coord_Type Propogator Start_time Stop_time
Step_Size(s) Coord_Sys Orbital_epoch Semi-major_axis(m) eccentricity
inclination arg_perigee RAAN true_anomaly'
%cmd = ['SetState */Satellite/LEOsat Classical J4Perturbation
"',scenario.StartTime,'" "',scenario.StopTime,'" 60 ICRF
"',scenario.StartTime,'" 6828140 0 45 0 0 0'];
% Better plan: Use the ephem data type to intialize the state vector.
% Following sat name: Coord_Type Propogator Start_time Stop_time
Step_Size(s) Coord_Sys Orbital_epoch X Y Z Xdot Ydot Zdot [all in m
or m/s]
cmd = ['SetState */Satellite/LEOsat Cartesian J4Perturbation
"',scenario.StartTime,'" "',scenario.StopTime,'" %d J2000
"',scenario.StartTime,'" 6828139.99999986 0.000488948476255031
-1.40040938458241 0.00110765265001805 5402.59868433944
5402.59801312843'];
cmdstr=sprintf(cmd,step);
%Run that command
root.ExecuteCommand(cmdstr);
%Propogate said satellite
LEOsat.Propagator.Propagate;
% Set the attitude of said satellite
root.ExecuteCommand('SetAttitude */Satellite/LEOsat Profile
XPOPInertial 0');
root.ExecuteCommand('AddAttitude */Satellite/LEOsat Quat "26 Sep 2016
12:00:00.00" 0.0 0.0 0.0 1.0');
% Add a sensor for payload and the TT&C
LEOsat_Pay = LEOsat.Children.New('eSensor', 'Payload');
root.ExecuteCommand('Define */Satellite/LEOsat/Sensor/Payload
SimpleCone 22.5');
LEOsat_TTC = LEOsat.Children.New('eSensor', 'TTC');
root.ExecuteCommand('Define */Satellite/LEOsat/Sensor/TTC SimpleCone
22.5');
root.ExecuteCommand('Point */Satellite/LEOSat/Sensor/TTC Fixed
Quaternion 0.0 0.7071 0.0 0.7071')
% Use a model to make it look pretty
root.ExecuteCommand('VO */Satellite/LEOsat Model File "C:\Users
\GhostStalker\Documents\MATLAB\DyLoMMot\12U AFIT Bus.dae"');
root.ExecuteCommand('VO */Satellite/LEOsat ScaleLog 3.7');
```

Calculate Access to MC3 Network and Export

```
root.ExecuteCommand('Access */Satellite/LEOsat/Sensor/TTC */Facility/
AFIT/Sensor/Sensor1 TimePeriod UseScenarioInterval')
root.ExecuteCommand('Access */Satellite/LEOsat/Sensor/TTC */Facility/
SDL/Sensor/Sensor2 TimePeriod UseScenarioInterval')
root.ExecuteCommand('Access */Satellite/LEOsat/Sensor/TTC */Facility/
NPS/Sensor/Sensor3 TimePeriod UseScenarioInterval')
root.ExecuteCommand('Access */Satellite/LEOsat/Sensor/TTC */Facility/
UHawaii/Sensor/Sensor4 TimePeriod UseScenarioInterval')
```

Generate target list

```
for i = 1:13
```

```
T{i} = scenario.Children.New('eTarget',['T' num2str(i)]);
end
%Spread the targets out (for a real mission, you'd want to put them in
real places)
setpos1 = ['SetPosition */Target/T1 Geodetic 20 -10 Terrain'];
setpos2 = ['SetPosition */Target/T2 Geodetic 11 0 Terrain'];
setpos3 = ['SetPosition */Target/T3 Geodetic 13 17 Terrain'];
setpos4 = ['SetPosition */Target/T4 Geodetic 25 15 Terrain'];
setpos5 = ['SetPosition */Target/T5 Geodetic 30 35 Terrain'];
setpos6 = ['SetPosition */Target/T6 Geodetic 33 40 Terrain'];
setpos7 = ['SetPosition */Target/T7 Geodetic 31 42 Terrain'];
setpos8 = ['SetPosition */Target/T8 Geodetic 33 44 Terrain'];
setpos9 = ['SetPosition */Target/T9 Geodetic 32 49 Terrain'];
setpos10 = ['SetPosition */Target/T10 Geodetic 35 53 Terrain'];
setpos11 = ['SetPosition */Target/T11 Geodetic 38 57 Terrain'];
setpos12 = ['SetPosition */Target/T12 Geodetic 40 59 Terrain'];
setpos13 = ['SetPosition */Target/T13 Geodetic 45 75 Terrain'];
%Run those set pos commands
root.ExecuteCommand(setpos1);
root.ExecuteCommand(setpos2);
root.ExecuteCommand(setpos3);
root.ExecuteCommand(setpos4);
root.ExecuteCommand(setpos5);
root.ExecuteCommand(setpos6);
root.ExecuteCommand(setpos7);
root.ExecuteCommand(setpos8);
root.ExecuteCommand(setpos9);
root.ExecuteCommand(setpos10);
root.ExecuteCommand(setpos11);
root.ExecuteCommand(setpos12);
root.ExecuteCommand(setpos13);
toc
```

Create and run command to generate LMMTfriendly .csv file of ground station access

```
rep_access = sprintf('ReportCreate */Satellite/LEOsat Type Export
Style "Access" File "C:\\Users\\GhostStalker\\Documents\\MATLAB
\\DyLoMMot\\AccessReportGS.csv" AccessObject */Facility/AFIT
AccessObject */Facility/NPS AccessObject */Facility/SDL AccessObject
*/Facility/UHawaii TimeStep %d', step);
root.ExecuteCommand(rep_access);
```

Set satellite payload to track targets when in view, then create target access report

```
point_targets = ['Point */Satellite/LEOsat/Sensor/Payload
Targeted File "C:\Users\GhostStalker\Documents\MATLAB\DyLoMMot
\Targetlist.txt"'];
root.ExecuteCommand(point_targets);
% Set the attitude to slew to targets (sensor will move independently
of body of spacecraft unless this is specified)
root.ExecuteCommand('SetAttitude */Satellite/LEOsat Target On');
for i = 1:13
root.ExecuteCommand(strcat('SetAttitude */Satellite/LEOsat Target ADD
Target/T',num2str(i)));
end
root.ExecuteCommand('SetAttitude */Satellite/LEOsat Target Slew Mode
FixedRate RateMagnitude 5 SlewTimingBetweenTgts Optimal');
tar_access = sprintf('ReportCreate */Satellite/LEOsat/Sensor/Payload
Type Export Style "Access" File "C:\\Users\\GhostStalker\\Documents
\\MATLAB\\DyLoMMot\\AccessReportTarget.csv" AccessObject */Target/
T1 AccessObject */Target/T2 AccessObject */Target/T3 AccessObject
*/Target/T4 AccessObject */Target/T5 AccessObject */Target/T6
```

```
AccessObject */Target/T7 AccessObject */Target/T8 AccessObject
*/Target/T9 AccessObject */Target/T10 AccessObject */Target/T11
AccessObject */Target/T12 AccessObject */Target/T13 TimeStep %d',
step);
%Propogate said satellite
LEOsat.Propagator.Propagate;
root.ExecuteCommand(tar_access);
```

Run the Moon and Sun reports LMMT requires

```
root.ExecuteCommand(sprintf('ReportCreate */Satellite/LEOsat Type Save
Style "Lunar Vector J2000" File "C:\\Users\\GhostStalker\\Documents\
\MATLAB\\DyLoMMot\\moon.txt" TimeStep %d', step));
root.ExecuteCommand(sprintf('ReportCreate */Satellite/LEOsat Type Save
Style "Sun Vector J2000" File "C:\\Users\\GhostStalker\\Documents\
\MATLAB\\DyLoMMot\\sun.txt" TimeStep %d', step));
```

Run the attitude and ephemeris output bits that LMMT requires

```
root.ExecuteCommand(sprintf('ExportDataFile */Satellite/LEOsat
Ephemeris "C:\\Users\\GhostStalker\\Documents\\MATLAB\\DyLoMMot\
\Satellite1_Mission.e" Type STK CoordSys J2000 TimeSteps %d', step));
root.ExecuteCommand(sprintf('ExportDataFile */Satellite/LEOsat
Attitude "C:\\Users\\GhostStalker\\Documents\\MATLAB\\DyLoMMot\
\Satellite1_Mission.a" Details Quaternions CoordAxes J2000 TimeSteps
%d', step));
disp('Reports generated for Mission Scenario');
toc
```

Published with MATLAB® R2016b

Bibliography

[1] C. A. Chung, *Simulation modeling handbook : a practical approach*. CRC Press, 2004.

[2] "AGI - software to model, analyze and visualize space, defense and intelligence systems." [Online]. Available: https://www.agi.com/products/stk/. [Accessed: 02-Dec-2016].

[3] D. Y. Stodden and G. D. Galasso, "Space system visualization and analysis using the Satellite Orbit Analysis Program (SOAP)," *1995 IEEE Aerosp. Appl. Conf. Proc.*, pp. 369–387.

[4] H. M. Udell, "A CubeSat Mission Modeling Tool," Air Force Institute of Technology, Master's Thesis, 2015.

[5] B. A. Jewell, "Applying Model-Based Systems Engineering to CubeSats: A Tailored Approach for a Reusable State Analysis Tool," Air Force Institue of Technology, Master's Thesis, 2015.

[6] J. R. Loudermilk, "A Logic-Based Missiong Modeling Tool for Designing CubeSats," Air Force Institute of Technology, Master's Thesis, 2016.

[7] J. R. Wertz, D. F. Everett, and J. J. Puschell, Eds., *Space Mission Engineering: The New SMAD*. Hawthorne: Microcosm Press, 2011.

[8] "Capabilities & Services | SpaceX." [Online]. Available: http://www.spacex.com/about/capabilities. [Accessed: 14-Jan-2017].

[9] "RocketBuilder." [Online]. Available: https://www.rocketbuilder.com/start/configure. [Accessed: 14-Jan-2017].

[10] G. Richardson, K. Schmitt, M. Covert, and C. Rogers, "Small Satellite Trends 2009-2013," *Proc. AIAA/USU Conf. Small Satell.*, p. SSC15-VII-3, 2015.

[11] "Commerical Space Launch Schedule and Pricing." [Online]. Available: http://www.spaceflight.com/schedule-pricing/. [Accessed: 02-Feb-2017].

[12] R. Cobb, "No Title." Dayton, OH, 2017.

[13] Y. A. Feldman, "FINITE-STATE MACHINES," *Encyclopedia of Computer*

Science and Technology. Marcel Dekker, Inc., pp. 73–104, 1992.

[14] "File:Automata theory.svg - Wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/File:Automata_theory.svg. [Accessed: 15-Jan-2017].

[15] R. M. Bond, "Project Insight: Threat Modeling And Assessment For Earth-Orbiting Satellites," Air Force Institute of Technology, Master's Thesis, 2006.

[16] J. E. McCarty, "A Simulink Based Tool for Designing Reference Mission Modeling," Air Force Institute of Technology, Master's Thesis, 2010.

[17] B. Andrews, "A Colony II CubeSat Mission Modeling Tool," Air Force Insitute of Technology, Master's Thesis, 2012.

[18] A. Hatch, "Electrospray Propulsion Interface and Mission Modeling for CubeSats," Air Force Insitute of Technology, Master's Thesis, 2012.

[19] "STK SOLIS - Advanced Solutions, Inc.," 2015. [Online]. Available: http://www.go-asi.com/solutions/stk-solis/. [Accessed: 02-Jan-2017].

[20] "New STK/SOLIS Software Delivers Mission, Trajectory and Spacecraft Simulation in One Commercial Product." [Online]. Available: http://vegas.digitalmedianet.com/article/New-STK/SOLIS-Software-Delivers-Mission-Trajectory-and-Spacecraft-Simulation-in-One-Commercial-Product--1723814. [Accessed: 02-Feb-2017].

[21] M. Swartwout, "The First One Hundred CubeSats: A Statistical Look," *J. Small Satell.*

[22] J. Loudermilk, "LMMT - How to Use Manual." 2016.

[23] G. Haun, W. Boudo, E. Swenson, D. Meyer, and J. Sadowski, "16 November 2016 Discussion between AFIT and AGI Personnel," 2016.

[24] "Cameo Systems Modeler." [Online]. Available: https://www.nomagic.com/products/cameo-systems-modeler. [Accessed: 17-Jan-2017].

[25] "Cameo Simulation Toolkit." [Online]. Available: https://www.nomagic.com/product-addons/magicdraw-addons/cameo-simulation-toolkit. [Accessed: 17-Jan-2017].

[26] “STK - MATLAB Interface.” [Online]. Available: http://help.agi.com/stk/index.htm#matlab/matlab.htm. [Accessed: 02-Feb-2017].

[27] “Chapter 3: CubeSat SE Example.” NASA, 2008.

[28] G. Prinsloo and R. Dobson, *Solar Tracking*. Stellenbosch: SolarBooks, 2015.

[29] S. A. Jacklin, “Survey of Verification and Validation Techniques for Small Satellite Software Development,” pp. 1–20, 2015.

[30] D. Kaslow, “CubeSat Model-Based Systems Engineering (MBSE) Reference Model.”

[31] “Systems Tool Kit (STK) Free.” [Online]. Available: http://www.agi.com/downloads/products/product-literature/120213_STK_Free_Flyer_Missiles.pdf. [Accessed: 06-Jan-2017].

[32] “Simulink - Simulation and Model-Based Design.” [Online]. Available: https://www.mathworks.com/products/simulink.html. [Accessed: 06-Jan-2017].

[33] “AGI - software to model, analyze and visualize space, defense and intelligence systems.” [Online]. Available: https://www.agi.com/. [Accessed: 06-Jan-2017].

[34] “MATLAB - MathWorks - MATLAB.” [Online]. Available: https://www.mathworks.com/products/matlab.html. [Accessed: 06-Jan-2017].

[35] “State Machine - Stateflow - Simulink.” [Online]. Available: https://www.mathworks.com/products/stateflow/. [Accessed: 02-Dec-2016].

[36] “STK - Connect: Command Listings.” [Online]. Available: http://help.agi.com/stk/11.0/Subsystems/connectCmds/connectCmds.htm. [Accessed: 19-Jan-2017].

[37] “MATLAB/Simulink - State Action Types.” [Online]. Available: https://www.mathworks.com/help/stateflow/ug/state-action-types.html. [Accessed: 19-Jan-2017].

[38] E. Swenson, “ASYS 632 Good Satellite Practices.” Dayton, OH, 2016.

[39] C. Alf *et al.*, “CubeSat Satellite Design,” Wright-Patterson AFB, 2016.

[40] G. E. P. Box, “Robustness in the strategy of scientific model building,” *Army Res. Off. Work. Robustness Stat.*, pp. 201–236, 1979.

VII Vita

Graduating from the University of Kansas in 2012, Capt Sadowski earned a Bachelor's of Science in Aerospace Engineering and a commission through the Air Force Reserve Officer Training Corps program. Alongside this research, he is pursuing an in-residence master's degree in Space Systems from the Department of Aeronautics and Astronautics at AFIT. He is scheduled to continue his career in the United States Air Force by working at the National Reconnaissance Office following the successful completion of his degree.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

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

TITLE AND SUBTITLE

Dynamic Logical Mission Modeling Tool

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Sadowski, Justin A., Captain, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/ENY)
2950 Hobson Way, Building 640
WPAFB OH 45433-8865

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GSS/ENY/17-290

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

10. SPONSOR/MONITOR'S ACRONYM(S)

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

12. DISTRIBUTION/AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A. APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This thesis research evaluates the usefulness of dynamic modeling as compared with static modeling (such as that which is already possible with the LMMT). When changes occur which make the spacecraft's method of interacting with the environmental model no longer relevant, due to either spacecraft mode changes or other agents in the simulation, the model should be dynamically updated to include these changes by means of repropagating the environmental model.
This research's focus is to ideate and then evaluate a subset of use-cases that would create changes in the environmental model. Through this research, it is hoped to develop a method for identifying when a static model such as the LMMT should be utilized versus a dynamic model such as that developed specifically for this research.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Eric Swenson, PhD, AFIT/ENY ADVISOR
U	U	U	UU	98	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-3636, ext 7479 (eric.swenson@afit.edu)

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18