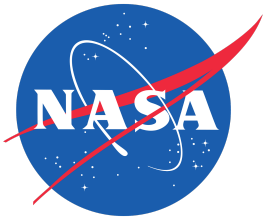
CougSat Simulation Software
Test and Validation Plans

Mentor
Aaron Crandall

Team Viking
Courtney Snyder
Angie Park
Edward Kuo
Solomon Egwuonwu

Course: CptS 423 Software Design Project II
Instructor: Aaron S. Crandall

TABLE OF CONTENTS

I. Introduction

I.1. Project Overview

The objective of this report is to assess and give an overview of our CougSat Simulation Software. The ultimate goal of the project is to build a simulator to test the behavior of CougSat I from launch to orbit, and ensure a successful mission and coherence to NASA's CubeSat standards. The major functionalities for which testing is crucial are contained in the attitude, communications, main, payload, and power boards. Currently, we have modelled each board's behavior as a state machine node in the Python library Transitions, and this semester will integrate those nodes into STK and simulate the satellite's behavior with the environmental triggers of space.

I.2. Test Objectives and Schedule

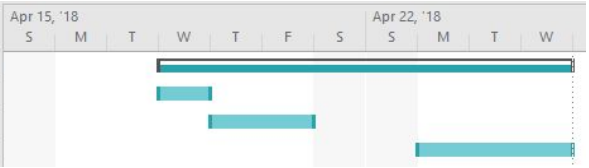
The approach to our testing involves the use of STK to build a model of the CougSat I and use this model as a testing tool to ensure that the inner-satellite boards behave as they should in space. We also want to confirm that CougSat I can communicate with the ground station(s) when it is in range. The inner-satellite boards are attitude, communications, main, payload, and power. In order to adequately test the simulator, we create various scenarios to accommodate all possible test cases that the CougSat I could encounter during orbit. As of now, the major resources required for this software is the AGI STK tool.

The major work activities are building the simulator, locating various ground station, creating various testing scenarios, running unit testing, integration testing and finally systems testing.

At the end of the semester, we will deliver working simulator, an accurate model of CougSat I, documentation and instructions to use the software, charts, and simulation results.

Our major milestone at the end of the semester is to produce a working simulator to help determine the behavior of CougSat I in launch and orbit

Task Name	Duration	Start	Finish
Final Testing	**6 days**	**Wed 4/18/18**	**Wed 4/25/18**
Unit Testing	1 day	Wed 4/18/18	Wed 4/18/18
Integration Testing	2 days	Thu 4/19/18	Fri 4/20/18
System Testing	3 days	Mon 4/23/18	Wed 4/25/18



I.3. Scope

The scope and purpose of this document is to give an overview of the CougSat Simulation Software and also to provide an overall approach to testing process, which include testing strategy and testing plans. In order for our software to be successful, we need to accurately represent the physical model and behavioral models of CougSat I and then successfully integrate those models into STK, which will provide the physics necessary to simulate the environment of low earth orbit. It is crucial for our software to represent the satellite as closely as possible so that we can provide Cougs in Space with the most accurate launch conditions

and ways in which the satellite can be improved to increase the likelihood for a successful mission.

II. Testing Strategy

Our software is required to test for valid communications within the satellite as well as from the satellite to the ground station(s). Each board will be tested individually, and each functionality of the board will be tested.

Attitude: The attitude board will detumble the satellite, position the satellite such that it is pointing at the sun, and report the satellite's position in latitude/longitude.

Communications: The communications board will communicate with the ground station whenever possible; it will transmit a "ping" to let the ground station(s) know that the satellite is still alive and functional, and it will receive commands from the ground station(s).

Main: The main board will essentially run the satellite; it will control the inner-satellite communications by being connected to each of the other boards.

Payload: The payload board will support the functionality that Cougs in Space decides to give it. We were not given details about the payload board, such as what the payload is, but we included it in the state machine to allow future users to include that functionality at a later time.

Power: The power board will control the power collection and distribution in the satellite.

This semester we decided to transfer our state machines from ROS Kinetic (C++) to Transitions (Python). This will allow us to include the satellite behaviors into the simulation in our script instead of in a separate file, making it more convenient and customizable for our client. The ROS unit tests we ran last semester verify that the nodes behave as they should when they do not have an environment acting upon them. Since the state machines were translated to Python, the black box test results should still hold true. We will integrate our state machine into STK and use that software to accurately simulate space. After this integration, we will need to retest each unit, as the environment in STK will have forces acting upon each node.

First, to determine if the attitude board will successfully detumble the satellite within STK, we will continue to use the method implemented in the attitude node. This method was running current through the x, y, and z coils, confirming that the satellite's respective yaw, pitch, and roll are all 0. Currently, we are looking into the method that STK provides to simulate detumbling and determining whether that or our state machine more closely represents CougSat I's detumbling strategy, so this test may need to be repeated if changes are made.

Next, to confirm that the attitude board can successfully position the satellite such that it is pointing at the sun, we will check that the solar panel on the top of the satellite is getting a maximal amount of sun and that the antenna is pointing at the earth.

Finally, to determine if the attitude board will successfully report the satellite's position in latitude/longitude, we will "stop" the satellite in-orbit within STK so that STK can report its position in latitude/longitude. Then, we will ask the satellite itself to report its position in

latitude/longitude and see if that value matches the value from STK. We will do this above each ground station to further confirm.

To determine if the communications board will communicate with the ground station whenever possible, it will transmit a “ping” to let the ground station(s) know that the satellite is still alive and functional. If the communication is successful, the ground station will receive the ping from CougSat I

We will also confirm that the communications board can receive and decrypt commands from the ground station(s). To do this, we will send multiple commands from the ground station and see if the satellite executes those commands. Ideally, we will execute one command on each circuit board.

The main board will essentially run the satellite; it will control the inner-satellite communications by being connected to each of the other boards. To test that it is working correctly, we will test intra-board commands like the ground station asking the satellite for its position (ground station -> communications board ->main board -> attitude board -> main board -> communications board -> ground station).

To determine if the power board will successfully control the power collection, we will see if the satellite is in the correct power mode. To test it, as there is no indicator showing the power mode, we will check what boards are on and make sure those boards are right to be on with the current power mode when the certain amount of the battery percentage is left.

To confirm that the power board will successfully distribute power throughout the satellite, we will set the initial battery level to some percentage and allow the power board to determine what power mode the satellite should be in while it orbits and collects more sunlight. We will then record how long the satellite can operate with that amount of initial battery level.

As we run each of these unit tests, we will thoroughly document the test data, test cases, test configuration, and test results. This information will help us improve the accuracy of our CougSat I model as well as providing constructive criticism for the components of the satellite that are accurate and failing in the environment of space. This documentation will also help us determine where bugs form and how they can be fixed.

Before the end of the semester, we will meet with members of Cougs in Space to familiarize them with the software. This will include helping them setup this environment on their machines, showing them how to make adjustments to the satellite behaviors using the Transitions library, teaching them how to change the physical satellite model using the Blender interface, explaining the concept of an STK scenario, explaining how to launch the program with our Python script, explaining varying launch coordinates and optimal ground station locations in that script, and finally leaving them with ample documentation to help new users setup and use or alter our software as necessary.

The documentation we provide Cougs in Space will be documents written by us and links and/or screenshots of references we used to set up our machines. We will support documentation for Windows 10, Linux Mint, and Ubuntu 16.04.

III. Test Plans

III.1. Unit Testing

As a simulator, our software provides a realistic behavioral and physical model of CougSat I as well as the environment it will be launched and orbiting in. To test our software, we will begin with unit testing. We began this process by implementing each circuit board as a state machine in ROS Kinetic and confirming that they behaved correctly with no environment and we translated them to Python. We also have developed the script file for STK simulator. Our unit testing will entail individually testing them. We will make sure that the state machines still accurately portray their respective boards and Python plug-in script file successfully simulates them while it is running.

What unit testing requires as the first action is splitting up units to test from the entire software. The unit testing on our software will be conducted with a several circuit boards. As of now, the CougSat Simulation Software has the following boards; Attitude, Communications, Main, Payload, Power, and Ground Station. The current version of the script can run STK based on the variables users set. However, for the unit testing, the script will only contain the solid variables using default values.

In Fall 2017, we implemented the boards as ROS nodes using ROS Kinetic. The unit tests we ran last semester verify that the ROS nodes behave as they should when they do not have an environment acting upon them. Since the state machines were translated to Python, the black box test results should still hold true.

However, we still have the state machines for the boards that indicate the behaviors the boards are supposed to achieve and we would be able to conduct the unit testing on the basic functions of each boards. Our plan for the unit testing is to completely isolate the boards, white box test them, and find any unexpected behaviors or bugs.

III.2. Integration Testing

Next, we will integration test each of the nodes with respect to each other, and the communication node with respect to the ground station(s). Again, this was briefly explored within the satellite last semester by running the ROS nodes concurrently and using the publisher/subscriber model. This was to ensure that the nodes communicated properly and interacted when the right circumstances occurred with no outside environment influencing these behaviors.

This semester, we will make sure that the nodes not only work together, but that they work together within the environment of space, and that the satellite can communicate with the ground station(s) when it is in range.

The integration testing respects the interaction between two or more boards. Based on designs from Cougs in Space and our implementation, these interactions happen when the Main board communicate with the other boards. The Python state machines behave as objects and communicate directly with the Main board, without the need for publishers and subscribers..

The ground station is also highly involved when it comes to the integration testing. It will communicate with the satellite through the radio, located on the communications board. The plan is to extend the integrating events to test. For example, the main node would like to check if the satellite is turned on. To do this, the main node checks if there is a current through each of the boards. Then, the main node will check if the satellite finishes the detumbling, which requires a response from the attitude node. We will compose the rest of the tests in this fashion. We will step forward to the next step with larger integration unless the test encounters the malfunctioning in the interaction for more than one component.

III.3. System Testing

III.3.1. Functional testing

Finally, we will system test the software as a whole. This will include launching the simulator using a Python script with a Python CONFIG file to allow users to change launch, orbit, or satellite preferences.

Testing the solar panels to see if the features of sleep and rotation of the panels towards the sun will operate as planned. This will make sure that the satellite will be able to continuously have power to transmit signals to the ground stations when it passes by.

III.3.2. Performance testing

Our requirements on performance will be whether the satellite is able to transmit signals back to the ground station. In addition to checking to see whether the satellite is able to transmit signals, the functionality of the components will be tested as well. This could be the speed of pictures sent from the satellite to ground station, the picture quality of the camera or the storage or accumulation of power from the solar panels.

III.3.3. User Acceptance Testing

For general user acceptance testing, the test will pass as long as it can transmit signals to the nearby ground station. The signals should be able to send and receive large packets in order to eventually support sending pictures taken by the satellite to the ground stations. In addition to testing for signals, we will need to make sure that the satellite will properly behave in orbit. This means working as a standalone satellite and have enough power to operate in orbit, with or without sunlight. The satellite should also be able to launch into space from a location specified in the CONFIG file.

We will confirm with the team leads and executive board that our software meets their requirements and is relatively easy to set up, use, and edit as the CougSat design evolves.

IV. Environment Requirements

The environment requirements of testing include multiple ground stations in which the satellite will be able to orbit above. The main ground station will be in Pullman, the alternate ground

stations will be in the other WSU Campuses. These include Tri-Cities, Spokane, Everett and Vancouver. The equipment on board the satellite will be a transmitter and receiver on the communications board to ping the ground station(s), and Cougs in Space hopes to have a camera to send pictures to the ground station(s). In addition to a camera, there will be up to 6 solar panels installed to turn sunlight into energy for the satellite. There will be a battery to hold the energy from the sunlight. The software we will be using is STK and Python. The ground stations should equipped with equipment that is able to transmit signals to and from the standalone satellite's antenna while it is orbiting Earth. This includes radio signals. The camera will be sending pictures to the ground station(s) through this method. The satellite will also be in an environment where it can be bright or dark to test the energy settings. These will be sleep and not sleeping. In the non-sleeping mode, the satellite should be expected to rotate and tilt the solar panels towards the sun.

V. Glossary

CONFIG file: A file used to configure the parameters and initial variables for the Python script that launches STK and creates CougSat I and ground station(s).

CubeSat: A cube-shaped satellite commonly used by researchers to collect data. They come in various sizes; 1 U, 2 U, 3 U, and 6 U. Each U is 10 cubic centimeters. CougSat I is in 1U cubesat.

Detumble: Stop uncontrollable tumbling of a spacecraft after launch.

Python Transitions: An object oriented state machine library in Python.

ROS Kinetic: Robot Operating System. An operating system that is useful for creating state machines and simulations. Kinetic is one of its different versions, and is the one we used in our alpha prototype.

STK: Systems Tool Kit. A satellite simulation tool made by AGI that allows users to customize diverse configurations for the satellite, spacecraft, ground facilities and more. Using it, we can build up the simulation for the model of CougSat I and run it for the testing.