CSULA Senior Design Project

2021-2022

Aerospace University Partnership Program

Applied Software Technologies Department

We propose an effort to explore technologies in the areas of satellite simulation, anomaly detection with machine learning techniques, and data processing. Here we provide a loose set of requirements that can be negotiated to either increase or reduce the challenges to the students as appropriate.

We encourage and expect the CSULA and Aerospace teams to talk about new and modified requirements frequently to optimize the benefit to everyone involved.

Aerospace Liaisons contact:

Denny Ly: [denny.ly@aero.org](mailto:denny.ly@aero.org)

Karina Martinez: [karina.martinez@aero.org](mailto:karina.martinez@aero.org)

Andre Chen: [andre.t.chen@aero.org](mailto:andre.t.chen@aero.org)

Vivian Sau: [vivian.sau@aero.org](mailto:\vivian.sau@aero.org)

# Project Overview

Satellites perform a host of vital functions including communications, weather prediction, geolocation, defense, and many others. In these complicated systems, it is extremely important that accurate data flows freely between the ground and the satellite via uplinks and downlinks. When strange behaviors or anomalies occur, it is vital that the error be identified and corrected before a disaster occurs. Sometimes these anomalies are the result of errors in the hardware or software, issues introduced by the environment, or an attack by a hacker. Effective anomaly detection techniques can help identify problems on the vehicle before they happen, which can help improve mission success.

The operation of satellites in long-term term operation is affected by many uncertain factors. Anomaly detection based on telemetry data is a critical satellite health monitoring task that is important for identifying unusual or unexpected events. The use of simulation tools allows users to configure and deploy platforms to be used in real time environments as well as simulate any anomalies that can take place. Machine learning can be used to detect these anomalies by comparing actual observed values with the predicted intervals of telemetry data. Simulation tools can be utilized by students to develop a way to solve these complex problems using applications already being used in the industry.

For this project, students will develop software components to integrate with and utilize existing industry open source software components to perform the tasks outlined below to:

1. Generate satellite simulation data
2. Inject anomalous scenarios into the flight system
3. Apply techniques for detecting the anomalies onboard and on the ground
4. Apply techniques for resolving anomalies onboard and on the ground

Expected outcomes from the project:

1. Software source to developed anomaly injection, detection, and resolution capabilities
2. Detailed documentation on design, implementation, tests, and results from each of the anomaly scenarios
3. User manual to set up, configure, and run the OSK with the anomaly injection, detection, and resolution capabilities
4. Monthly review meetings with Aerospace liaisons and final outbrief to Aerospace engineers

# Software Tools and Components

The project will leverage [OpenSatKit (OSK)](https://github.com/OpenSatKit/OpenSatKit), which combines three powerful open source tools that are currently used in real missions today: Ball Aerospace Corporation's [COSMOS](https://cosmosrb.com/) ground system, NASA Goddard’s [core Flight S](https://cfs.gsfc.nasa.gov/)ystem(cFS) flight software, and NASA Goddard’s [42](https://software.nasa.gov/software/GSC-16720-1) satellite simulator. See the documentation for OSK in its [GitHub Wiki](https://github.com/OpenSatKit/OpenSatKit/wiki).

Each major software component is described in more detail in the sections below.

## Core Flight System – Flight Software

OSK provides a complete desktop solution for learning how to use NASA's open source flight software (FSW) platform called the core Flight System (cFS). The cFS is a reusable FSW architecture that provides a portable and extendable platform with a product line deployment model. The cFS has significant flight heritage, provides a complete set of command and data handling functions required by most spacecraft, and is very reliable. A virtual environment with OSK set up will serve as the development environment for learning about flight and ground system communications as well as providing a platform for developing anomalies to be injected into the simulation. OSK comes with the cFS preconfigured for a fictitious satellite called SimpleSat (SimSat).

## 42 – Spacecraft Simulator

In addition to cFS, OSK uses NASA Goddard’s 42 dynamic satellite simulator for simulated hardware command and telemetry. 42 is a comprehensive general-purpose simulation of spacecraft attitude and orbit dynamics. Its primary purpose is to support design and validation of attitude control systems. 42 accurately models multi-body spacecraft attitude dynamics as well as modelling environments from low Earth orbit to throughout the solar system. It also features visualization of spacecraft attitude.

## COSMOS – Ground System

OSK implements extensive COSMOS configurations and customizations so COSMOS can serve as the primary OSK user interface. COSMOS is a suite of applications that can be used to communicate with the satellite, monitor its performance and health, and display its data. The systems that COSMOS interfaces with can be anything from test equipment (power supplies, oscilloscopes, switched power strips, UPS devices, etc), to development boards (Arduinos, Raspberry Pi, Beaglebone, etc.), to satellites.

COSMOS implements a client server architecture with the Command and Telemetry Server and the various other tools typically acting as clients to retrieve data. The Command and Telemetry Server connects to the targets and sends commands and receives telemetry (status data) from them. Targets are the items you’re trying to control or get status from.

# Anomaly Injection, Detection, and Resolution – Flight Software

Traditionally, satellites use basic sensors and thresholds on board to detect anomalies. For example, if a value from a temperature sensor goes too high, then a flag may be raised and the operators on the ground may also see the issue in their computers and software. The limitation here is that more complex issues will be missed by basic thresholds (i.e. power consumption increase within a certain threshold may be OK when the satellite is commanded to perform an operation, but not when it is idling), but they can be detected with more sophisticated analysis.

By configuring and using OSK, we have total control over the flight software and the data that is produced in the simulation. The main task for this project will be to leverage the software components to (1) inject anomalous behavior/data, (2) use modern data analysis techniques/libraries/frameworks to detect those anomalies where traditional threshold checks would fail, and (3) automate the resolution of the anomalies detected.

Anomaly detection capabilities can be implemented in two areas, onboard the satellite and on the ground system. The advantage of onboard processing is that the detection software will have direct access to all the sensory data on the satellite. An anomaly that causes a failure in transmission might mean the operators on the ground never get to see the problem. Being closer to the actual hardware also allows for earlier detection, which in turn provides more time to respond to the anomaly. The downside is limited resources available onboard. Typical onboard processors have limited memory and CPU cycles running on a single thread. Therefore, any anomaly detection software will consume CPU cycles that could otherwise be used for performing real work. An alternative solution can be a separate anomaly detection hardware that is flown with the payload, but again resources are limited.

The other opportunity for anomaly detection is by doing it on the ground. Computing resources on the ground are nearly unlimited, so more advanced detection capabilities can be applied. Limitations are reduced response time due to satellite contact time, data payload size restrictions when downlinking, as well as opportunities to respond to anomalies due to scheduling. Satellite contact time depends on the orbit the satellite is in and the number of ground stations, potentially limiting contact with the satellite to a couple times a day for less than an hour. Payload limitations are also in place due to bandwidth speed to transmit data to the ground, so only high priority telemetry data is made available. Finally, response can be further delayed based on priority of commands that are being scheduled to be sent up to the satellite on the next contact.

After an anomaly is detected, onboard processes and/or ground processes are executed to try to resolve the anomaly. For example: data storage on the spacecraft can experience data corruption, so sometimes “error correcting code” (ECC) memory is used, which can automatically detect and correct certain types of data corruption. The operators on the ground might be alerted (by the spacecraft) that data corruption was detected and corrected. However, in situations where an anomaly cannot be resolved automatically on-board the spacecraft, the operators on the ground might need to manually execute an anomaly resolution process on the ground (perhaps requiring the upload of a software patch, or requiring that the satellite be put into “safe mode” while other anomaly resolution processes take place). Manual processes can be error-prone and slow, resulting in incorrect or delayed anomaly resolution. Automated anomaly resolution, both on-board and on the ground, is desirable.

As you can see, there are a lot of factors that come into play when developing solutions for anomaly detection. More focus will be placed on onboard processing, with ground processing playing a secondary role. Anomalies to be injected will be required to be developed but should be trivial once a good understanding of OSK and its interfaces has been established. Some examples of possible anomalies include runaway tasks, memory leaks, denial of service, invalid command sequences, and single bit errors.

# Requirements

## Development and Testbed Environment

## Objectives

Provide a flight software, ground station, and spacecraft simulator testbed environment for developing anomaly injection, detection, and resolution capabilities.

## Top Level Requirements

* + - 1. Team shall establish a development environment with OSK installed and configured to be used for learning about OSK and as a testbed to test anomaly injection, detection, and resolution.
         1. Each team member shall have their own development environment set up to enable parallel development and testing efforts
         2. OpenSatKit shall be deployed and used for communication between all components (cFS, COSMOS, 42).
      2. Team shall establish a ground system with COSMOS configured to communicate with the flight system.
         1. Ground system shall be able to send commands to the space system
         2. Ground system shall be able to downlink telemetry data from space system
         3. Ground system shall deploy ground-based anomaly detection and notification capabilities.
         4. Ground system shall deploy automated anomaly resolution capabilities.
      3. Team shall establish a flight system with cFS and 42 configured and communicating with the ground system.
         1. Flight system environment shall have restricted CPU and memory resources to be determined at a later date.
         2. Onboard anomaly detection and notification capabilities shall be deployed. Once anomaly is detected, event reports shall be generated and queued for next contact with ground.
         3. Onboard anomaly resolution capabilities shall be deployed.

## Anomaly Injection

## Objectives

Develop various anomalous scenarios along with a capability to inject these scenarios into the flight system.

## Top Level Requirements

* + - 1. Team shall develop anomaly injection capabilities to include, but not limited to, the following scenarios:
    - component failure (affecting sensor, actuator, thruster, solar panel, thermal, star-tracker, and other systems)
    - unexpected halt and/or reboot of main processor
    - multiple single-event upsets (SEUs) occurring in short period of time
    - loss of communication with ground

## Anomaly Detection

## Objectives

Develop the capability to detect anomalous behaviors during the mission and generate notifications when anomalies are detected.

## Top Level Requirements

* + - 1. Team shall develop onboard anomaly detection and notification capabilities
         1. Anomalies detected shall generate event logs and queued for next contact with ground.
      2. Team shall develop ground-based anomaly detection and notification capabilities.
         1. Anomalies detected shall generate event logs as well as notify analysts of the event.

## Anomaly Resolution

## Objectives

Develop the capability to automatically resolve anomalies after they have been detected.

## Top Level Requirements

* + - 1. Team shall develop onboard anomaly resolution capabilities.
         1. Spacecraft shall automatically resolve anomalies via its onboard anomaly resolution processes, if possible.
         2. Spacecraft shall inform ground of any anomalies it is trying to resolve, the status of the resolution, and any additional information that ground requires for successful anomaly resolution
      2. Team shall develop ground-based anomaly resolution capabilities
         1. Ground shall have the capability to automatically resolve anomalies detected onboard the spacecraft