2.7 Using simulation

Practical guidance – space

Authors: ACTIONS demonstrator project

Simulations are vital to assuring space systems, whether autonomous or otherwise. Unlike with many other robotic systems, the operating environment of a spacecraft is not easily accessible for testing. Space is both costly to access for satellites and other spacecraft and, once launched, spacecraft are rarely recoverable such that system updates and re-deployment can occur. This raises the challenge whereby:

- Spacecraft must be effectively complete prior to launch, as updates after launch range are typically either costly or impossible
- The environmental stimuli required to fully test the spacecraft are unavailable prior to launch, leading to incomplete testing

The former challenge can be partially met through the use of standardised mechanical, electrical and software designs and components, and the ability to update the software post-launch. The use of a post-launch commissioning process facilitates this and is standard for the majority of missions. The latter can be met using simulation, of which there are several approaches available for space systems. This guidance focusses on the specific case of satellite systems and missions. Satellite missions include one or more satellites (the space segment) and one or more ground stations for telecommanding and data reception (the ground segment).

Simulation for space

There are many existing solutions for simulating satellite systems and missions. These typically fall into three categories:

- Mission-level: High-level mission analysis and planning tools
- Platform-level: Orbital and attitude simulators
- Operational: Simulators for conducting rehearsals and tests of mission operations

Mission analysis and planning tools such as NASA's General Mission Analysis Tool (GMAT) [1], Gpredict [2] and Systems Toolkit (STK) [3] are used to simulate single satellites and constellations over time periods ranging from a single orbit (~90 mins in low Earth orbit) to several months. This is typically used to analyse how often the space segment of a mission will interact with both the ground segment and other ground targets such as regions or features of interest. It can be used to determine how long each satellite spends in eclipse (darkness) or illumination (sunlight), which affects the power budgets of the mission due to a satellite's typical dependence on solar power. It can be used to evaluate how such aspects evolve over the course of the mission, whether due to change in season or due to degradation of both satellite altitude and systems. Such tools do not typically include lower-level modelling of sensors and actuators. They are used to aid in designing the mission on a high-level, allowing determination of parameters such as orbit properties, number of satellites, number and location of ground stations and high-level satellite requirements such

as solar panel surface area and battery size. They can also be used to aid mission planning. After launch, tools such as GMAT and Gpredict can be used to predict a satellite's position and determine exactly when it will be contactable from a ground station.

A level down from this are orbit and attitude simulators. These are often used to simulate a single satellite over a shorter period of time, from a section of orbit to several successive orbits. The orbital path will be the same as that predicted in mission-level tools, but lower-level aspects such as actuators, sensors, power systems and environmental disturbances can now be modelled with higher fidelity. This allows hardware components to be specified and modelled in simulation before development begins. It allows testing of software in real-time, from low-level actuator controllers to single satellite activity scheduling and even constellation management. Many industrial companies have their own simulation platform for such purposes, often built on either open-source tools such as 42 [4] or using toolboxes in software such as MATLAB / Simulink [5]. It is common that such simulators are interfaced with physical hardware to create hardware-in-loop (HIL) or system-in-loop (SIL) simulators, where as much of the actual satellite system as possible is used. Simulation then provides only those elements that are impossible to recreate on the ground, such as orbital and attitude mechanics and some aspects of sensors and actuators.

Finally, operational simulators exist for evaluating the behaviour of satellite components when interfaced with others. These can be purely software-based [6] but may also be HIL or SIL, with simulation used to supply components with realistic telemetry. These simulators are commonly known as "flatsats", as the usual satellite component stack is laid out flat to facilitate easy integration and revision of components [7]. In addition to facilitating hardware testing, the flatsat also enables operational rehearsals, as the integration of real flight software and telemetry and telecommand components allows human operators to command and control the satellite in real-time. Use of a coaxial attenuated radio link can further increase the realism of the simulation, with the radio switched on and off as needed to emulate satellite passes over the ground station. Integration of operational simulators with platform-level simulators can enable real-time rehearsals and testing with realistic telemetry and environmental stimuli. The use of visualisation to verify commands is also beneficial here, such as showing the satellite rotate in response to actuator commanding.

Integration of all three simulation types is possible. STK and MATLAB can be integrated with each other to enable HIL simulation through STK, enabling higher-fidelity simulation in STK and the use of realistic ephemeris data in operational rehearsals and testing.

Application to autonomous systems

The use of autonomy in satellites is by design meant to enable more responsive mission operations, whether those operations involve tasking on-board instruments and control systems, communicating with other satellites and ground stations, or processing, managing and delivering data acquired in-orbit (such as optical or radar data). It is possible to classify satellite autonomy along many taxonomies (system criticality, level of human involvement, etc.) however, the classification which determines the use of simulation testing on autonomous systems is whether the system is open- or closed-loop. These two types are summarised in Table 1.

Table 1 – Comparison of open-loop and closed-loop satellite autonomy, with examples.

Classification	Description	Examples
Open loop autonomy	Autonomous decision making does not affect the critical operations of the satellite (or other satellites). It is open loop in the sense that these decisions do not affect what the satellite is sensing or how it otherwise interacts with the ground (such as ground station passes). The behaviour of the satellite is therefore predictable over a given period of time. Operational decisions made on the ground by human operators may be informed by on-board decisions, but this not considered part of the autonomous loop and is not fundamentally different from traditional satellite operations.	Data management, e.g. tagging and queuing of data based on inferred value. Data product creation, e.g. alerts, masks and images created from raw payload data.
Closed loop autonomy	Autonomous decision making on-board the satellite affects its critical operations, typically through tasking of the platform actuators and sensors, the payload and communications systems (whether for communicating with a ground station or other satellites). The behaviour of the satellite is therefore dependent on what the satellite is perceiving at any given time and is highly dynamic with respect to the timescales involved in mission-level simulations such as STK.	Instrument tasking to re-acquire targets identified on previous passes. Actuator control, typically to point sensors at specific ground or orbital targets. Comms management, autonomously transmitting data, including commands to other satellites.

The strengths and weaknesses of each simulation type are then evaluated against the two autonomy categories in Table 2 below.

Table 2 – Summary of impacts of using simulation tools with autonomous satellites and overall strengths and weaknesses.

Simulation Type	Impact of Open-Loop Autonomy	Impact of Closed-Loop Autonomy
Mission-level	Little impact as mission operations still involves human operators and are therefore predictable and deterministic on the time scales used in such simulation tools. Statistical models can be used in lieu of realistic sensing environment models.	The high-frequency decision making of an autonomous system means that satellite behaviour may be unpredictable on the timescales used in such simulations. Activities such as autonomous attitude control may be difficult to model. Statistical models may suffice for some activities such as instrument tasking or ground station and intersatellite communications.
Platform-level	Simulation must include sensing environment models of sufficient fidelity to synthesise payload data for ML component inference. These could be pre-generated data loaded from memory.	Sensing environment models are again required, and data must be generated in real-time to account for satellite state changes. Native support for HIL, flight software and actuator controllers is otherwise sufficient for autonomous systems.
Operational	Sensing environment data is again required but can be pre-generated. Operational commanding will likely use higher-level commands aggregate more traditional platform instructions	Sensing environment data is again required but can be pre-generated for a range of different operational scenarios which cover the known operating conditions of the satellite. Operational commanding will again involve higher-level commands, for example providing mission goals or targets rather than low-level actuator, payload and mode commands.

Simulation testing in the ACTIONS project

The AAIP demonstrator project, ACTIONS, explored a relatively simple application of autonomy to satellites. A mission was designed which enabled autonomous detection of wildfires from orbit, and then the reporting of these fires and their locations to end users on the ground. On-board machine learning was used to enable the identification of wildfires in spatial data acquired by the satellite's optical instrument payload. This scenario was therefore an example of open-loop autonomy, with decisions made on-board affecting only the management of data and the creation of data products.

As a result, the satellite behaviour on a high-level remains predictable over the timescales considered in mission-level simulations. As such, GMAT was used to provide an initial estimate of the constellation size required to meet re-visit frequency requirements for a defined region of interest on the ground, in this case the state of Oregon.

A customised platform-level simulation was created using an open-source simulation tool as a basis. This was extended with an instrument and sensing environment model which enabled the input of real satellite image data into the hardware-deployed payload data processing chain. Near real-time alerts for fires were then autonomously generated from this data and validated against ground truth data, providing assurance for the autonomous software. The architecture for this platform simulation is shown in Figure 1.

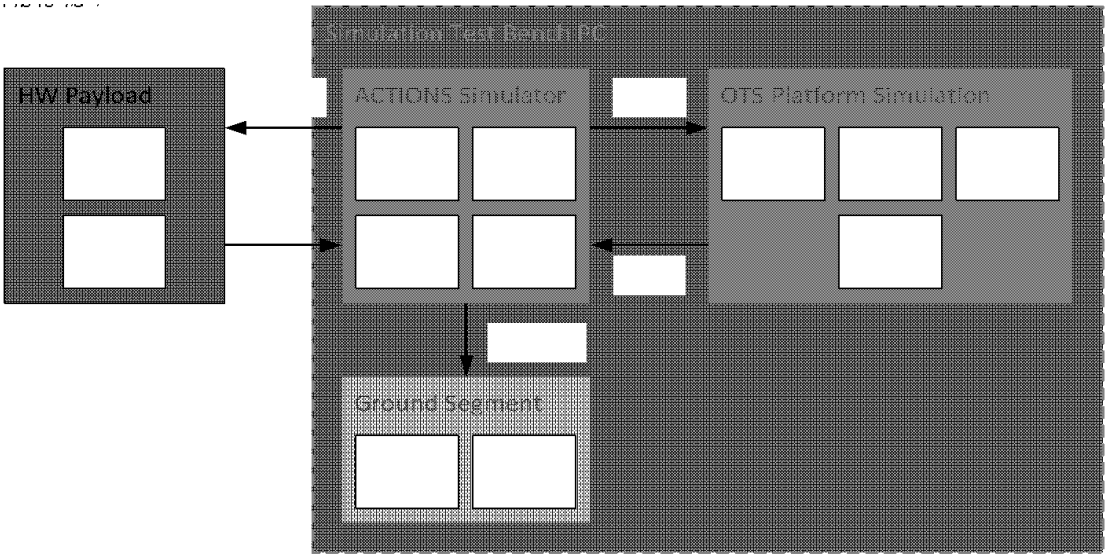

Figure 1 – Architecture of ACTIONS platform simulation, augmenting a standard platform simulation with a custom engine which supports interface with hardware components, an instrument and visual environment model, simple attitude controller and a basic ground segment for receiving and visualising data products.

The use of a visual environment model in ACTIONS is critical; instrument payload data is the key driver of autonomy in the satellite and ultimately supplies the main output of the mission: data products and emergency alerts. Satellites often use multispectral instruments which preclude the use of simple computer-generated environments, which are often used in other simulation tools such as AirSim [7]. These instruments capture multiple spectral bands, typically ranging from the blue to the shortwave infrared parts of the spectrum. As satellite instruments must typically capture unbroken "swaths" of data, the rate at which they capture data is thus related to their speed with respect to the ground and the spatial extent that can be captured in a single acquisition. In ACTIONS, a contiguous swath of image

frames over Oregon required a new frame to be captured every 5 seconds. The key benefits of the simulation were therefore:

- The use of real, multispectral satellite imagery of Oregon, which could be transformed to emulate the data acquired by the instrument payload and used to verify and validate the data products created by the autonomous processing chain. This is stored locally on the test bench PC.
- The use of realistic, real-time simulated telemetry (time, position, velocity and attitude) to synthesise instrument data from the Oregon image dataset.
- The provision of the simulated telemetry to the processing chain, which is used to calculate the ground coordinates of acquired image frames and the wildfires identified within them. This is known as geolocation.
- Close coupling of satellite orbital physics and processing chain latencies, where an image frame must be fully processed, and its products stored at least as fast as the instrument is acquiring new data (which is then based on constant geometries and orbital velocity).

A screen capture of the ACTIONS simulation, showing a 3D animation of the satellite in Earth orbit (for visualisation only) and the red-green-blue (RGB) bands of the synthesised multispectral image, is provided in Figure 2.

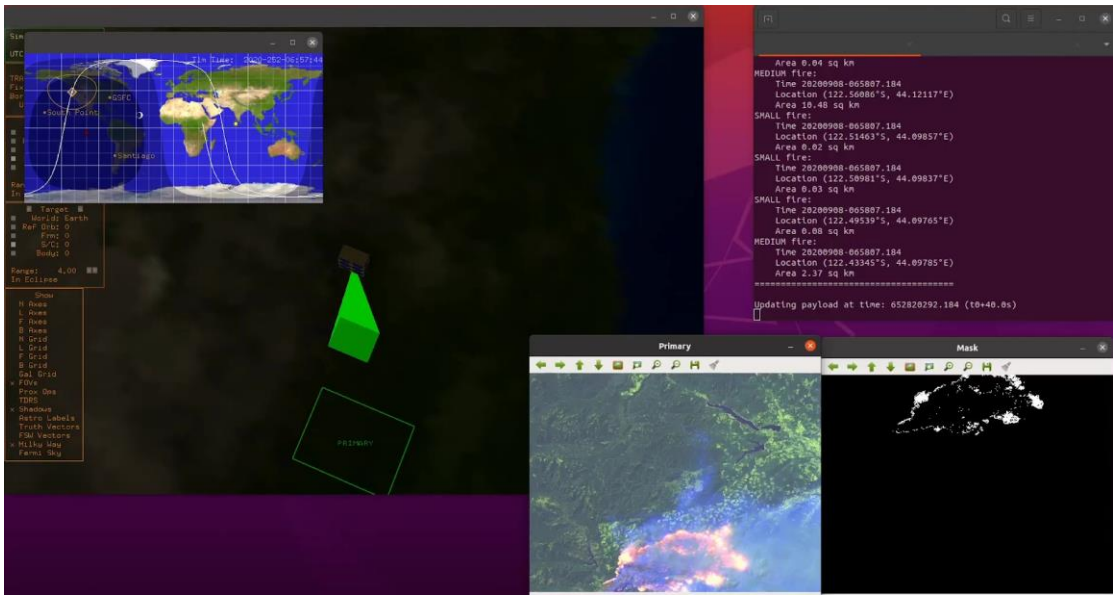

Figure 2 – Screen capture of ACTIONS simulator, showing 3D orbital animation, satellite ground track, RGB visualisation of multispectral image capture, fire pixel mask from prior data acquisition and terminal output with plain text fire alert information.

The safety requirements of the ACTIONS mission related to the timeliness of fire alerts, the spatial accuracy of wildfire geolocation and the accuracy of fire predictions by the ML component (both false positive and false negative). The platform-level simulation then allowed the following tests to be performed:

- Evaluation of the speed of the end-to-end processing chain against the acquisition rate of the instrument, including the machine learning component and other tasks.
- Validation of the accuracy of machine learning predictions of wildfires (and wildfire-free regions) against the truth data in the simulation image dataset.

- Validation of the geolocation precision of the generated data products against the truth data.
- Evaluation of the overall improvements to data management and throughput (the "end benefits") of safety-critical data and the ability to meet system-level safety requirements.
- Evaluation of the impact of anomalies in synthesised instrument data on inference accuracy and the end benefits of the autonomous processing chain.

An operational simulator was not used for ACTIONS, as the operation of the autonomous data processing chain is automatic following the acquisition of an image frame by the simulated satellite instrument. In reality, this acquisition can easily be driven by a spacecraft activity schedule operating on a fixed frequency. As the platform simulator includes hardware-in-the-loop, an operational simulation would add little value to testing at this stage.

Outcomes of testing

The outcomes of simulation testing for ACTIONS can be summarised with respect to generic autonomous satellite testing as follows:

- Existing platform simulation tools can be readily augmented or extended to support the necessary visual environment models for the *sensing* element of a SUDA (or SUDA-style) autonomous system.
- Platform-level simulation on real-time timescales is critical to evaluating the impact of sensed environmental data (whether visual, inertial, geometric, etc.) on real-time autonomous systems.
- Platform-level simulation tools can be used to evaluate specific operational scenarios over shorter periods such as a single orbit or section of orbit.
- Mission-level validation of safety requirements can be performed using labelled or easily verifiable simulation test datasets.
- Mission-level simulation tools such as STK and GMAT can be used to design and plan full missions, even autonomous ones, provided appropriate statistical models are in place to represent high-frequency autonomous behaviours. These can model single or multiple autonomous satellites over periods from days up to years.
- Faults and anomalies can be readily simulated and their impact evaluated in platform-level and operational simulations.

Summary of approach

1. Before development, use mission-level simulators to design and plan mission, augmenting with suitable statistical models to abstract autonomous behaviours
2. Develop and deliver software and relevant hardware for autonomous satellite(s)
3. Integrate hardware-deployed autonomy software (and any other relevant flight components/software) into platform-level simulation
4. Validate behaviour and performance of autonomy software in real-time simulation for multiple test scenarios, including nominal and anomalous conditions and edge cases

References

[1] *GMAT*. Available: https://sourceforge.net/projects/gmat/files/GMAT/GMAT-R2016a/.

[2] *Gpredict.* Available: http://gpredict.oz9aec.net/.

[2] 'Systems Tool Kit (STK)', *Agi*. https://www.agi.com/products/stk.

[3] E. T. Stoneking, *42 - Spacecraft Simulation*. 2022. Available: https://github.com/ericstoneking/42

[4] 'MATLAB', *MathWorks*, https://uk.mathworks.com/products/matlab.html.

[5] M. Asbury, 'NASA Operational Simulation for Small Satellites', NASA, Aug. 22, 2017. http://www.nasa.gov/centers/ivv/jstar/nos3.html.

[6] 'FlatSat — a ground-based testbed for CubeSats at ESA/ESTEC', *eoPortal*. https://directory.eoportal.org/web/eoportal/satellite-missions/content/-/article/flats-1.

[7] 'Home - AirSim', *Microsoft*. https://microsoft.github.io/AirSim/.