SSC23-XIII-06

NASA Operational Simulator for SmallSats (NOS3) – Design Reference Mission

John P. Lucas
NASA IV&V, Jon McBride Software Test and Research (JSTAR)
100 University Ave., Fairmont, WV 26554
John.P.Lucas@nasa.gov

Matthew D. Grubb, Justin R. Morris, Mark D. Suder, Scott A. Zemerick
NASA IV&V, Jon McBride Software Test and Research (JSTAR)
100 University Ave., Fairmont, WV 26554

ABSTRACT

The NASA Operational Simulator for Small Satellites (NOS3) has undergone significant advances including updating the framework to be component based and expanding the open-source code to include a generic design reference mission to enable advanced technologies. This paper details the changes to the framework as well as a number of innovative use-cases the team is currently supporting such as 1) the expansion of NOS3 to support distributed systems missions in collaboration with NASA GSFC, 2) the integration of NASA JPL's Science Yield improvemeNt via Onboard Prioritization and Summary of Information Systems (SYNOPSIS) for on-orbit science data prioritization, and 3) the inclusion of NASA IV&V JSTAR's software-only CCSDS encryption library (CryptoLib). NOS3 continues to serve the SmallSat community by providing an open-source digital twin that can significantly reduce costs associated with spacecraft software development, test, and operations. The NOS3 team plans to continue to expand the resources available to the community and partner with others to resolve issues and add new features requested via the NASA GitHub.

INTRODUCTION

The NASA Independent Verification and Validation (IV&V) Program's mission is to provide assurance that safety and mission-critical software will operate reliably and safely. NASA IV&V provides this service by employing a set of documented technical methods to the customers' system and software requirements, design, code, and tests. In 2009, the NASA IV&V Program established a simulation development team, the Jon McBride Software Test and Research (JSTAR) Team [1]. The JSTAR team is responsible for developing, integrating, and maintaining digital-twin test environments that are capable of exercising mission and safety critical software to increase mission assurance. This capability thus enables the NASA IV&V Program to perform more thorough analyses of software behaviors, unit, build, and system level software, tests and operational test procedures.

JSTAR developed digital-twin software-only simulation environments have enabled risk reduction testing, provided earlier execution of operational tests, reduced the development organization's reliance on hardware, and increased available test resources on large spacecraft missions such as Global Precipitation Measurement (GPM), James Webb Space Telescope (JWST), and the Nancy Grace Roman Space Telescope [2]. In addition to these large missions, the JSTAR team has applied its technologies to small satellites, which suffer from some of the same challenges such as long hardware lead times and limited software development/testing resources.

As a result of the demonstrated successes of high-fidelity digital-twin environments and the opportunity to launch a spacecraft to demonstrate technologies that benefit NASA programs through the CubeSat Launch Initiative (CSLI), the NASA IV&V Program and West Virginia University (WVU) collaborated to develop a 3U SmallSat mission, Simulation-to-Flight-1 [3].

The STF-1 mission resulted in the development of a digital-twin software simulation framework named the NASA Operational Simulator for Small Satellites (NOS3). The goal of NOS3 is to enhance small satellite software development, testing, and training while making the framework open source for the community. With NOS3, the flight software executes as if it were operating in space. NOS3 provides the flight software with representative real-world simulated data inputs that it would expect during nominal on-orbit operations. Some of NOS3 features include:

1) enabling multiple developers to build and test flight software with simulated hardware models

2) serving as an interface simulator for science instrument / payload teams to communicate with prior to hardware integration
3) supporting software development activities
4) enabling hardware integration to parallel software development
5) providing automated testing framework
6) increasing available test resources
7) enabling operation of the simulated spacecraft using the ground software command and telemetry databases

NOS3 OVERVIEW

NOS3 integrates a set of existing open-source software modules as well as JSTAR developed middleware to create a full spacecraft simulation platform. Figure 1 illustrates how these individual software components are interconnected within NOS3 to create a baseline deployment. Additional details on each of these software modules and the other configurations supported can be found in [4], but by default NOS3 utilizes the core Flight System (cFS) [5] flight software, COSMOS ground software, and 42 as the dynamics engine. NOS3 is conveniently packaged as a ready-to-run virtual machine, reducing the overhead associated with installing and configuring the framework.

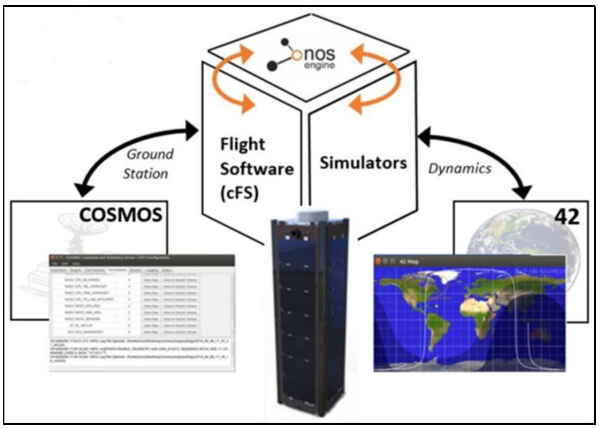

Figure 1. NOS3 Module Interconnections

NOS3 was used extensively by the STF-1 software development team for all aspects of flight software development and testing. Over the course of three months in which most STF-1 software development was accomplished, each team member used their own NOS3 digital-twin as their development and testing environment. The virtual environment provided realistic inputs and feedback to the flight software while under development and enabled additional testing by being open source and requiring no license purchase or setup to use.

Thanks to NOS3, the STF-1 SmallSat mission software development and testing was completed significantly ahead of schedule and STF-1 launched on schedule in December 2018. At the time of this writing STF-1 has been on-orbit for over 1600 days and remains operational and healthy. Zero flight software bugs have occurred on-orbit.

COMPONENT DEVELOPMENT

Since STF-1 was launched, NOS3 as a framework continued to be updated and cloned for use in new missions. Minor changes to support new host operating systems and update packages were the primary focus as maintenance was simply done in engineers' free time. When NOS3 was used in support of a recent development effort from GSFC that included multiple missions with similar or identical hardware, it was realized that improvements to NOS3 needed to be made to enable a more plug-and-play configuration with the hardware options available to each mission.

New missions continued to request various small hardware changes that initially appeared simple but affect multiple repositories and require a large amount of effort. For example, if a new magnetometer was used then this would impact not only the flight software application responsible for communicating with the component, but also the attitude control system to subscribe to these new messages and interpret them. Additionally, the ground software database would need to be updated for the new commands and telemetry available to an operator and the dynamics engine would require parameter updates. Also, the new magnetometer would require a new hardware simulator to simulate its behavior such as the use of a different bus protocol and different byte level interchanges. Each of these sets of source code and parameters were stored in a different repository and maintaining the changes was a major hurdle.

Realizing these hurdles, NOS3 was modified to what has been called a component based framework where all code related to that physical hardware is captured in a single repository. This code includes the flight software, the hardware simulator, and the ground software database. This removes the need to synchronize various repositories together to ensure compatibility and enables different mission teams to reduce rework by having a single reference where patches and new releases are readily applied. To reduce the startup energy a component template and development flow was also established.

The component development flow starts with using the template generator. The template provides a standard format between components which can lead to easier

updates when API changes are required in NOS3. The second step is a component documentation review. The component readme is then updated with critical information on the software interface between the hardware component and the flight software including which documents and versions were utilized and a test plan is drafted. At this point it is recommended that another developer or integration and test team member also review the documentation and the completed readme prior to starting code development.

Once the readme and test plan have been updated as necessary and any clarifications required with the vendor are resolved, the development of a standalone checkout application can begin. This checkout application can be built to run in the NOS3 simulation or on a development board through build flags passed into the NOS3 hardware library. If the hardware is available, it is recommended to continue to develop and test the checkout application. Hardware availability is often an issue and in that case the simulation can also be developed and tested with the checkout application directly. Note that the simulation is not a replacement for traditional hardware testing, but an additional tool to be used to reduce schedule and risk.

As was the case for many mission hardware components during the pandemic, standalone checkout application testing may finish prior to the availability of hardware. The flight software application can be developed utilizing the same functions and hardware library calls used in the standalone checkout. This means that the flight software application is primarily an integration test with the rest of the software – including the ground software and the associated integration tests documented in the test plan. Additional time should be allocated to update the component after hardware testing does occur.

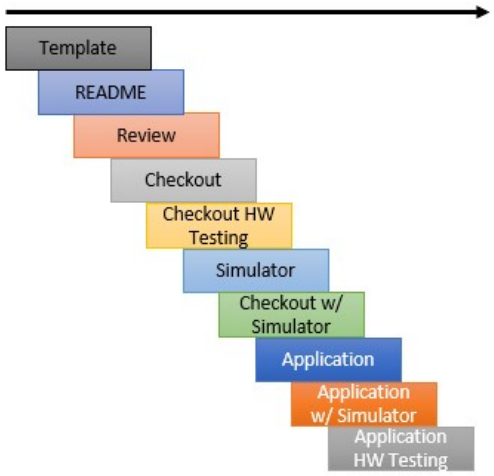

Figure 2. NOS3 Component Development

DESIGN REFERENCE MISSION

The latest mission users of NOS3 have also provided feedback that more generic examples and training needs to be a priority. Completed missions often become so specific that demonstrating to possible users requires an understanding of all the underlying software modules included. To address this, NOS3 was updated from a framework to a design reference mission. This provides a starting point to build up documentation on how the flight software operates and would be controlled by the operator via the ground software in addition to develop new advanced technologies on.

NOS3 release 1.6.1 includes generic components that would be used on a standard mission. These include a camera, coarse sun sensor, electrical power system, fine sun sensor, magnetometer, GPS, inertial measurement unit, radio, reaction wheels, and torquers. Through these generic component examples, standard commands and telemetry that differ from the template were captured and related to the specific type of actuator or sensor. In addition, the various hardware bus interfaces currently supported now have a reference for future development. The establishment of this baseline is enabling NOS3 to tackle standardizing the interfaces between these generic component types to the flight software applications that require sensor fusion such as attitude determination and control systems. This is an ongoing development at the time of writing.

Specifics such as the startup sequence of the reference spacecraft, where that occurs in code, and other examples or known modifications are being compiled to be included in the wiki of NOS3. Through the process of documenting these specifics the team has realized that configuration control of the mission needs to be the next major step in development. While components are now modular, the system itself is still highly coupled to achieve the desired goals.

FUTURE NOS3 EFFORTS

Continued development of NOS3 has sparked interest in additional support. Support comes both directly in deploying NOS3 for specific missions, adding new features, and integrating external software to the baseline. The development team has made efforts in the work with NASA GSFC and NASA JPL which will be available to the community. These various efforts are detailed in the following subsections.

Distributed Systems Missions

Recently there has been a movement toward distributed systems missions (DSMs). These mission systems are comprised of various combinations of spacecrafts in a constellation, ground stations, satellite relays, rovers,

drones, etc. To enable further research, NASA GSFC has funded continued development to support such missions in NOS3. To accomplish this the NOS3 team has staffed up and moved to do all development, road mapping, and issue tracking on the open-source GitHub. This will enable additional feature requests and input from the community on the direction NOS3 is taking. GSFC is supporting other advanced technology development that will be integrated and demonstrated using NOS3.

The first hurdle NOS3 has chosen to tackle is that of multiple spacecraft in a constellation. This poses problems for scalability and configuration management. A simple scenario of multiple of the exact same spacecraft in a string of pearls configuration has been selected as the initial target with a single ground station. Initially a demonstration was performed using three virtual machines.

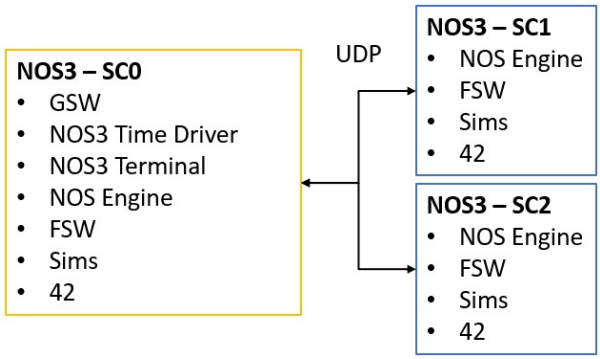

Figure 3. NOS3 DSM Proof of Concept

The primary machine was named SC0 and housed all the standard modules seen with a standard single NOS3 deployment with the exception that the time driver configuration was modified to send time ticks to the other machines as well. SC1 still had its own flight software, middleware with NOS Engine, simulators, and dynamics engine provider as did SC2. A basic demonstration of communication between spacecraft was also performed by triggering a relative time sequence set of commands on SC0 which then reconfigured the sample component and forwarded the desired configuration to the next spacecraft in the chain via the proximity communication channel of the radio. This exercise showcased additional limitations in the current architecture of NOS3, primarily the reliance on *localhost* for networking.

A second demonstration is being developed that utilizes containerization to repeat the above scenario and enable further comparison. While this development is not yet completed it has already been determined that the networking can be more easily resolved and the possibility of running directly from your host exists. Downsides include having to understand yet another technology in addition to the complexities of a spacecraft system to run. Should the team select this configuration after assessment, NOS3 would likely require updates to the means of deployment and helper scripts that are provided.

SYNOPSIS

Recent interest in NOS3 integrations at other NASA centers such as JPL has resulted in the current effort to integrate with the **S**cience **Y**ield improveme**N**t via **O**nboard **P**rioritization and **S**ummary of **I**nformation **S**ystem (SYNOPSIS). SYNOPSIS is a software package designed to significantly enhance the scientific return from interplanetary spacecraft missions. The challenges of scientific exploration, particularly beyond Mars, are amplified by the limited data bandwidth available for transmitting information over vast interplanetary distances. Traditionally, missions have been constrained to capture only as much scientific data as can fit within the allocated data bandwidth. However, with the increasing amounts of instrument data, advancements in onboard computing capabilities like the Snapdragon processor, and rationing of Deep Space Network (DSN) time due to ongoing and upcoming missions, a paradigm shift is now favorable.

SYNOPSIS offers a solution by leveraging onboard data processing to transition from a strategy of "collect some, return all" to "collect much, return best." By dynamically analyzing and ranking data using onboard science autonomy, SYNOPSIS aims to substantially improve the scientific return of interplanetary missions.

The SYNOPSIS framework, shown in Figure 4, operates on two core algorithms to process and rank data. First, it employs a prioritization algorithm that quantifies the utility of data based on its scientific value. This component calculates the similarity between collected instrument data and the high-value targets identified by the science team. For instance, in a life detection mission to Enceladus, the algorithm would prioritize data related to moving microscopic organisms [6] and indications of metabolic chemistry [7]. In the case of a water detection mission to Mars, the algorithm would prioritize data related to chemical compounds indicative of previous liquid water on the surface.

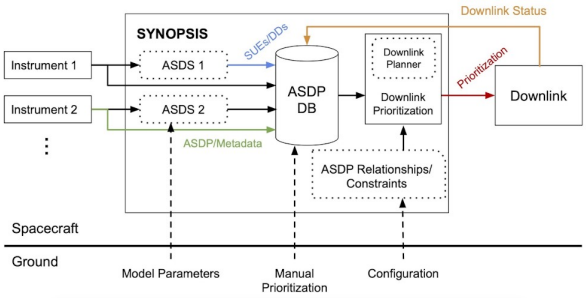

Figure 4. SYNOPSIS Spacecraft and Ground Architecture for Data Prioritization

Second, SYNOPSIS utilizes a set of rules and constraints that define relationships and restrictions across different instruments. For instance, if a ChemCam measurement on the Mars Science Laboratory (MSL) discovers a high-value chemical compound, the associated MastCam image acquired from the sample site would receive an increased priority. These algorithms are developed collaboratively with mission science teams and can be fine-tuned during flight using configuration files. With these powerful algorithms, SYNOPSIS generates a comprehensive and prioritized list of scientific data for one or more instruments on the same spacecraft, ensuring that the most valuable information is selected for downlink transmission.

Integration activities have been successful with SYNOPSIS into NOS3. Initially, SYNOPSIS is being integrated with the core Flight System flight software as a library inside NOS3, however, integration is being performed generically to support other flight software frameworks, such as JPL's FPrime [8]. SYNOPSIS will be open source in the future, and when it is, it will be included in the NOS3 baseline for use.

CryptoLib

CryptoLib was originally developed by the JSTAR team to support interoperability testing between the ESA and NASA; however, its continued development has been encouraged by both JPL and GSFC. The goal of CryptoLib is to provide a software-only solution to secure communications between a spacecraft and the ground. The Consultative Committee of Space Data Systems (CCSDS) Space Data Link Security (SDLS) standard is used to secure communications. A software only solution was utilized to best support having security be a plug and play module to existing architectures.

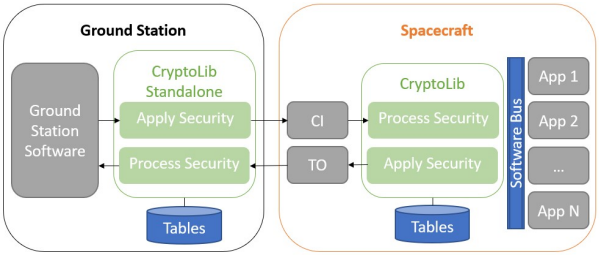

Figure 5. CryptoLib Overview

CryptoLib 1.2.2 was included in the NOS3 1.6.1 release. This release includes CryptoLib in both the spacecraft software and as a standalone interface in the ground software communication chain. Once the Advanced Multi-Mission Operations System (AMMOS) team releases the Key Management and Cryptography (KMC) software as open source this will be included in NOS3 instead of the standalone implementation for the ground software to provide a more user-friendly implementation. CryptoLib supports command authentication, encryption, and authenticated encryption to aid missions in meeting the NASA-STD-1006A requirements.

At the time of writing, CryptoLib supports the command link only. Development is underway to support the telemetry link and expand the library to be more modular. Having the library be more modular will enable other underlying FIPS libraries to be used to perform the cryptography while CryptoLib continues to support interpreting and enforcing the NASA standard.

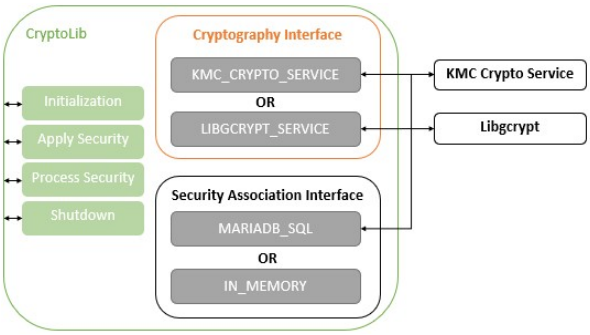

Figure 6. CryptoLib Interface Example

Work to support CCSDS SDLS Extended Procedures (SDPS-EP) is also planned and tracked in the open-source GitHub.

CONCLUSION

The NASA IV&V JSTAR team plans to continue development of NOS3 and integration efforts into next year. While development continues, the community is encouraged to provide feedback to help steer the activities.

REFERENCES

1. NASA. (n.d.). Jon McBride Software Test and Research (JSTAR). [Online]. Available: https://www.nasa.gov/centers/ivv/jstar/JSTAR.html. Accessed on: June 3, 2023.
2. S. A. Zemerick, "Development of Software-only Simulation Test Beds (SoST) for Spacecraft and SmallSats," Graduate Theses, Dissertations, and Problem Reports, 7888, 2020. [Online]. Available: https://researchrepository.wvu.edu/etd/7888.
3. Morris, J. et al. (2016): Simulation-to-Flight 1 (STF1): A Mission to Enable CubeSat Software-based Verification and Validation, presented at 54th AIAA Aerospace Sciences Mtg., San Diego, CA, US, Jan. 4–8. Paper 6.2016-1464
4. Geletko, D. M., Grubb, M. D., Lucas, J. P., Morris, J. R., Spolaor, M., Suder, M. D., ... & Zemerick, S. A. (2019). NASA Operational Simulator for Small Satellites (NOS3): the STF-1 CubeSat case study. arXiv preprint arXiv:1901.07583.
5. NASA. (n.d.). NASA core Flight System (cFS). [Online]. Available: https://cfs.gsfc.nasa.gov/. Accessed on: June 3, 2023.
6. Wronkiewicz, Mark & Lee, Jake & Mandrake, Lukas & Lightholder, Jack & Doran, Gary & Mauceri, Steffen & Kim, Taewoo & Oborny, Nathan & Schibler, Thomas & Nadeau, Jay & Wallace, James & Moorjani, Eshaan & Lindensmith, Chris. (2023). Onboard Science Instrument Autonomy for the Detection of Microscopy Biosignatures on the Ocean Worlds Life Surveyor.
7. S. Mauceri, S. Massie, and S. Schmidt, "Correcting 3D cloud effects in X_{CO2} retrievals from the Orbiting Carbon Observatory-2 (OCO-2)," Atmospheric Measurement Techniques, vol. 16, no. 6, pp. 1461-1476, 2023. [Online]. Available: https://amt.copernicus.org/articles/16/1461/2023/. DOI: 10.5194/amt-16-1461-2023
8. NASA F´/Prime, "Home," [Online]. Available: https://nasa.github.io/fprime/. [Accessed: June 3, 2023].