SSC22-WKIV-05

MO Services and CFDP in Action on OPS-SAT

Dominik Marszk, David Evans, Tom Mladenov, Georges Labrèche
European Space Operations Center (ESOC), European Space Agency (ESA)
Darmstadt, Germany; +49 615 190 2219
dominik.marszk@esa.int

Vladimir Zelenevskiy
Telespazio Germany GmbH
Darmstadt, Germany; +49 615 182 570
vladimir.zelenevskiy@telespazio.de

Vasundhara Shiradhonkar
Terma GmbH
Darmstadt, Germany; +49 6151 860050
vash@terma.com

ABSTRACT

OPS-SAT is a 3U CubeSat launched by the European Space Agency (ESA) on December 18, 2019. It is the first nanosatellite to be directly owned and operated by ESA. The spacecraft is a flying platform that is easily accessible to European industry, institutions, and individuals enabling rapid prototyping, testing, and validation of their software and firmware experiments in space at no cost and no bureaucracy. Conceived to break the "has not flown, will not fly" cycle, OPS-SAT has spearheaded many firsts in both space and ground segments. For instance, its uplink rate is four times higher than any ESA spacecraft; it employs never before flown communication protocols, and it implements new ESA patents. Proven and standard approaches to space operations are difficult to break away from in a sector that epitomizes risk-averseness. This is particularly true with how packet telemetry and telecommand are addressed using the Packet Utilization Standard (PUS). OPS-SAT has set aside PUS in favor of a standard that is currently being defined by the Consultative Committee for Space Data Systems (CCSDS), that is, MO Services and the File Delivery Protocol (CFDP)'s file-based operations. OPS-SAT is the first in-orbit demonstration of fully MO-based on-board software and ground implementations. With over 220 experiment proposals submitted, a robust file transfer and management system greatly reduces the complexity and cost of operating multiple on-board software instances that must reliably deliver results back to experimenters. This paper details the design, implementation, and operations of the MO Services and CFDP on OPS-SAT. It presents the benefits of developing, deploying, and maintaining the MO/MAL ground infrastructure with OPS-SAT as a case study. Lessons learned as well as recommendations from the spacecraft's flight-proven experience in adopting and operationalizing the standard are presented as invaluable feedback for future missions as well as for CCSDS ongoing development of the standard.

INTRODUCTION

OPS-SAT is an ESA nanosatellite mission designed exclusively to demonstrate ground-breaking satellite and ground control software under real flight conditions. This makes it the first mission of its kind worldwide. The project is led by the European Space Operations Center (ESOC) in Germany underlining it as a mission designed by operators for operators. OPS-SAT mission details and its history is explained in detail in [1, 2]. This paper describes how OPS-SAT leverages the latest CCSDS standards to simplify development, deployment, and operations within a complex, distributed multi-user space-ground environment. The spacecraft bus consists of an on-board computer called the NanoMind, a power subsystem, a UHF communications subsystem, and a basic ADCS subsystem. The Satellite Experimental Processing Platform (SEPP) is the heart of the OPS-SAT. It is a powerful system-on-chip (SoC) module with sufficient on-board memory to carry out advanced software and hardware experiments [3, 4, 5, 6, 7].

To provide context, this paper first introduces the details of the MO and CFDP standards then it explains applications implementing those standards. After which, the paper goes into the details of how the satellite and the

ground segment integrates those applications. A set of challenges and solutions at various stages of the implementation and integration of the stack are presented followed by outlines proposed as future work. Finally, a conclusion is given.

CCSDS MISSION OPERATION SERVICES

CCSDS MO is a set of standard end-to-end services based on a service-oriented architecture that is intended to be used for mission operations of future space missions. OPS-SAT has pioneered both its implementation and operationalization. MO Services is a layered framework that specifies mission operation services in a platform-independent manner that abstracts out low-level implementation and communication details. Interoperability between mission operation services across different framework implementations is made possible with the Message Abstraction Layer (MAL). The MAL defines a set of basic data types that are used as building blocks to construct messages for MO standard service operations [8]. Exchanged messages between ground and space segments are defined in an easily parsable common machine-readable format (XML).

MO standards are divided into:

- Services Areas
- Transport Binding and Encoding

MO Areas define groups of application services and their data models. The standardized MO areas include:

1. Message Abstraction Layer (MAL) — a special fundamental area defining only basic data types and interaction patterns, which all the other services can reuse.

2. Common Object Model (COM) — defines how structured data can be persisted (COM::Archive), various activities can be tracked (COM::ActivityTracking), and provides a generic event transport and persistence model (COM::Event). Most of the higher-level services build on top of the COM.

3. Monitoring & Control (MC) — provides facilities to define and invoke generic telecommands (MC::Action), define and monitor telemetry parameters (MC::Parameter), raise alerts (MC::Alert), define and control autonomous parameter value checks (MC::Check), provide statistical analysis of the telemetry (MC::Statistics), create parameter aggregations (MC::Aggregation), and define parameter calibrations (MC::Conversion).
4. Common Area (Common) — defines a set of support services, aiming to ease integration and implementation of MO-based systems. This includes a central directory for discovering other MO services (Common::Directory), authentication and authorization endpoint (Common::Login), and a generic service configuration persistence model (Common::Configuration).

All the above standards can be explored in an interactive web application available under [9].

There are other Areas currently under development, including:

1. Mission Planning and Scheduling (MPS) — aims to define a set of common interfaces for mission planners and plan executors.

2. Mission Data Product (MDP) — aims to define a set of common interfaces for requesting, managing, and retrieving data products.

3. File Management (FM) — defines a set of operations allowing to easily manage the file system and the file transfers, meant to complement the CFDP.

Transport bindings and encodings of MO specify how MAL messages can be transported over a set of standard protocols. Common transport bindings available include Space Packets, TCP/IP, ZMTP, and HTTP, while encodings include Binary, String, and XML.

In addition to the standard service areas, the system designers are free to define their own bespoke specifications similar to what is done for Mission-Specific PUS Services [10]. Such services can be easily integrated into applications based on the reference MO communication stack implementation, thanks to the open-source code generation tooling available for processing the interface descriptions.

The MO Services take as little assumptions about the system implementation and the deployment as possible. Thus, they are structured in a manner agnostic to the protocol used, communication method, and topology — i.e., Space-to-Space (S2S), Space-to-Ground (S2G), or Ground-to-Ground (G2G). Later it is shown that various subsystems of OPS-SAT flight and ground segment utilizes these unique characteristics.

CCSDS FILE DELIVERY PROTOCOL

The CFDP protocol ensures file transfer robustness with packet correctness and loss recovery. It is designed to allow deploying over a wide range of space missions, fitting different operational and connectivity scenarios. This is achieved by defining 2 main Put Modes or Classes [11]:

- Class 1 — unreliable, best effort delivery — i.e., Unacknowledged Mode.
- Class 2 — reliable, guaranteed complete file delivery — i.e., Acknowledged Mode.

In addition to the above, CFDP standard defines a variety of different options like negative acknowledgement (NAK) modes (Immediate, Deferred, Prompted, Asynchronous), Protocol Data Unit (PDU), cyclic redundancy check (CRC), Bounded or Unbounded files (with or without predefined size), Segmentation at Record Boundary, different Actions on Fault.

The CFDP standard books provide a very detailed description and guidelines for implementation of the transfer state machine. The transferred PDUs can be encapsulated over any packet or stream-based transport.

NANOSAT MO FRAMEWORK

The NanoSat MO Framework (NMF) was developed to provide a standard, open-source on-board software framework for nanosatellites that follows the MO services framework architecture. It facilitates monitoring and controlling the spacecraft's software applications as well as interaction with the platform and its peripherals. The NMF inherits all the advantages of the MO framework but with a special focus for nanosatellites with pre-implemented components that achieve a full end-to-end system when assembled to satisfy mission requirements. The NMF consists of a set of tools, core components, and APIs. The core components and APIs are defined as a set of "composites" that the applications can reuse and infer. The build products are deployed across the two segments: Ground and Space [12, 13, 14].

In addition to the standard MO services, the framework defines a set of bespoke services like Platform or Software Management, aiming to abstract away the complexity of interfacing with the platform payloads.

NMF is split into 2 main parts — core and mission tailoring components. NMF Core includes a simulator and its build product is a fully functional Software Development Kit, allowing the developers to create applications agnostic to the execution platform.

Notable parts of the Core, besides the basic building blocks for MO-compliant applications, are Supervisor and Ground MO Proxy. The Supervisor allows orchestrating the execution of every application by implementing the Software Management services provider, it also serves as a Platform service provider, through which all the application-to-payload interactions are performed. The Ground MO Proxy integrates into the Ground Data System, and performs transparent, opportunistic space to ground replication, serving as a base interface endpoint and a building block for a Monitoring and Control system for the remotely executed applications.

Mission tailoring components are then used on an actual satellite deployment in place of the simulation layer, to interface with the real hardware. The mission tailoring and the simulator are exposed to the applications via an abstraction layer of the Platform services, which is identical for all the deployments. This achieves a cross-platform and cross-mission compatibility of the applications built with the NMF.

The MO services implemented by the NMF can be explored in an interactive web application available under [15].

FILE MANAGEMENT SERVICES

The FMS concept is that both ground and space applications employ identical architecture, and each run its own MO FMS provider with its companion CFDP engine. An MO service consumer (i.e., client) can connect to either space or ground instance and request for a certain operation to be executed (e.g., create folder). If this operation requests for a file transfer, then the CFDP engine is activated, and the transfer is initiated between space and ground components. The client/consumer can monitor the transfer.

The FMS CFDP daemon is configured to operate in Class 2 (Acknowledged) mode with immediate NAKs.

DEPLOYMENTS ON OPS-SAT

There are 3 distinct and parallel deployments using MO services across space/ground within the OPS-SAT ecosystem. All can be independently used to interact with different subsystems of the spacecraft in a specialized manner.

On-Board Software

OPS-SAT On-Board Software (OBSW) employs an embedded, lightweight implementation of the MO services, using selected capabilities of MAL, COM, and MC as the Space-to-Ground and Space-to-Space interfaces between the OBSW, Ground Data System, and the payload computer.

Even though OBSW does not implement the COM Archive, it builds heavily on its model and the necessary definition details are provided in an offline manner to the control system, trading off part of the robustness coming from MO discoverable model, for much simpler and smaller implementation, which is elementary for an embedded system.

Not only OBSW employs a set of standard services, but also defines its own bespoke interfaces, like Power Management, to interface via the EPS, rudimentary File and Memory interfaces, allowing to quickly access all of the OBSW memories and even upload and apply binary delta patches of the OBSW image itself.

NanoSat MO Framework

NMF provides a full implementation of the MO standard, deploying most of the capabilities of MAL, COM, MC, Common service areas specifications. The OPS-SAT mission tailoring for NMF implements a set of platform specific adapters, on one side interfacing with the NMF Platform services implementation, and on the other with a set of native payload driver libraries. The adapters use the SWIG tool to automatically generate a glue layer between the native libraries and the Java Native Interface.

FMS/CFDP

FMS daemon is a specialized application where a set of bespoke MO Services (File Management Services) are used as a control layer for the CFDP endpoints deployed across Ground and Space. It does not contain much of OPS-SAT specific tailoring, besides configuration specific to the mission profile and a support for encapsulation of the CFDP PDUs in the CCSDS Packets. Furthermore, OPS-SAT SEPP is running 2 independent FMS daemons to support 2 redundant communication interfaces — one allowing to perform transfers over the CAN Bus with a downlink baud rate limited to 1 Mbps, and the other operating on the SpaceWire bus, allowing to achieve downlink baud rates up to 12 Mbps.

GROUND DATA SYSTEM

The mission operations concept is robust and at the same time low cost. The concept relies on automation as the primary mode of changing configurations. Experiments are uploaded by experimenters to a Data Relay Server and tested on an Engineering Model FlatSat at ESOC to check if they meet the basic safety requirement. Then they are uploaded to the spacecraft's eMMC memory using the File Management Services. Experimenter's data is compressed into a single file on-board, downloaded and distributed directly to them. The experimenters are provided with different interfaces with which they can access the ESA infrastructure to monitor and control their experiment in real-time or offline mode.

The reference diagram for interactions across the entire OPS-SAT system between the NMF and different space and ground components is shown in Figure 1.

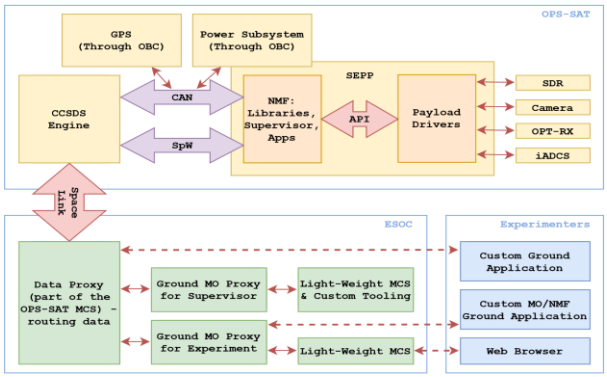

Figure 1: Reference application stack with example interactions on-board and to the ground.

OPS-SAT MCS Data Proxy

OPS-SAT Data Proxy is a ground component tightly integrated into the MCS, serving as a central multiplexer for all ground systems and tools, being able to connect to a variety of operational and test systems e.g., ground station modems, EGSE equipment, CCSDS Engine emulator, simulated sources and sinks of data. It is one of the core components separating the experimental and core operations, capable of routing and filtering data streams based on the CCSDS Packet header Application ID (APID). The Proxy can transparently interleave and convert streams at different encapsulation levels e.g., CCSDS Packets, TM Frames, CLTUs, and SCOS NCDU Frames. Furthermore, it is also an MO Service Provider, defining a set of bespoke operations that the MCS can issue on the TC link for the Proxy to intercept and execute. Data Proxy user interface is shown in Figure 2.

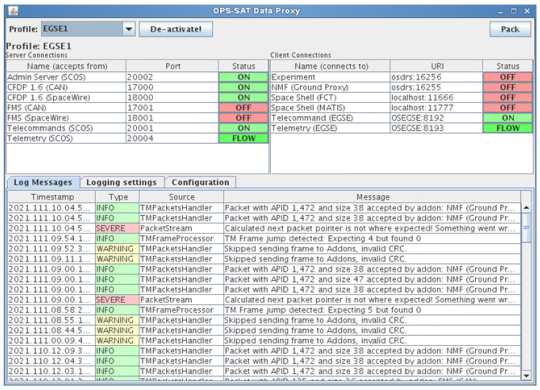

Figure 2: OPS-SAT Data Proxy interface.

Support for the OBSW MO Services

The ground segment is centered around the European Mission Control Software, SCOS-2000, which has been modified to handle the new application-level interface CCSDS MO Services and packets. The OPS-SAT MCS build and configuration system is equipped in a tool called MO MIB Generator, which ingests the XML interface description of the OBSW, FMS, and Data Proxy as part of a nightly Continuous Integration build to produce a fully functional SCOS-2000 MIB without any manual action needed. This is made possible by SCOS-2000 MCS's high configurability, which was originally developed for PUS.

ESA ground segment infrastructure tools that are necessary for OPS-SAT operations, such as the Mission AutomaTIon System (MATIS), can interact with SCOS without complication brought about MO integration. MO Operations and MC Actions are fully abstracted into the SCOS Telecommands model, which is the basis for interaction with the external tools and cleanly translates into MAL Space Packet Protocol PDUs in the same manner as it does for PUS.

Support for the NMF applications

To maximize the simplicity of operating the NMF applications, ESA has developed a Light-Weight Mission Control System (LWMCS) — an MO-native, M&C tool with a web UI, dedicated to interacting with standard MO applications over either a space-to-ground or ground-to-ground interface. The core functionality was largely inspired by the mostly used SCOS applications — Manual Stack and Parameter Display. LWMCS is deployed in OPS-SAT operational and development environments, allowing parallel operations on both flight and engineering satellite models. Furthermore, thanks to Docker containerisation, LWMCS binaries are provided to all OPS-SAT experimenters free of charge for in-house testing.

LWMCS itself has been built with modularity and compatibility in mind, supporting standard MO interfaces on both sides of its core component, allowing it to support a variety of GUIs and space-to-ground protocols. The high-level architecture of the LWMCS is shown in Figure 3.

The system utilizes the self-descriptiveness of the MO data model to automatically explore newly discovered remote providers (NMF applications) and synchronize to them, with the goal to provide to the user a seamless plug-and-play experience.

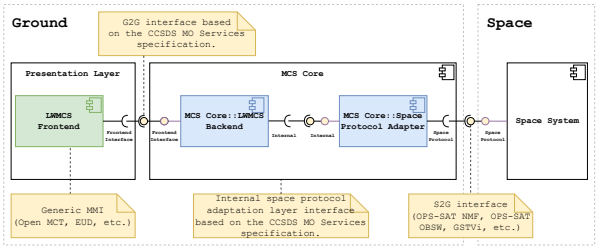

Figure 3: LWMCS Architectural Diagram.

Integration of the FMS/CFDP

Thanks to the implementation and deployment topology described in the *File Management Services* section, FMS has both ground and space components which are effectively the same software.

For OPS-SAT two simple Groovy scripts were implemented: upload and download. This means that users can execute upload/download files from the command line. Groovy was selected as it is supported by our automation system MATIS.

The ground segment FMS application is integrated into the ground system via the Data Proxy. OPS-SAT features two file transfer chains for CAN bus and SpaceWire transfers. This is also reflected on the ground, having two parallel FMS daemons. Their implementation is identical, and they differ only in configuration. The interactions between the FMS applications of a single file transfer chain are shown in Figure 4.

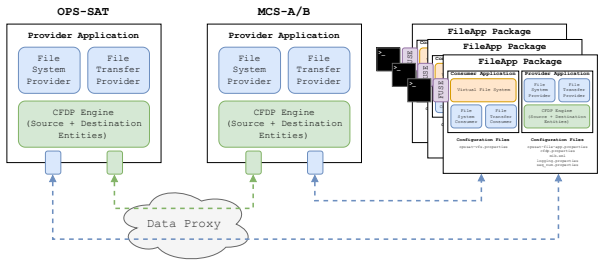

Figure 4: Interactions between the FMS daemons of a single file transfer chain.

CHALLENGES AND SOLUTIONS

The following section outlines notable obstacles encountered and solutions implemented when adopting the discussed applications.

MO Services

Most of the MO standards, when adopted by OPS-SAT, were still largely in draft phase. This accounted for some inconsistencies between the OPS-SAT OBSW interfaces and the published standards. Parts of the OBSW where this has caused problems were updated to the latest standard over time. The rest of the OBSW still operates using draft interface revision, while NMF has been updated to be in line with the published standard. To avoid a potential interfaces incompatibility, MO service version field, transmitted in every exchanged packet, has been utilized, changing the OBSW MO interfaces version annotation, from "1" to "127". This allowed the NMF to support and use 2 different versions of the same COM and MC interface binding — one for communicating with the ground, another for communicating with the OBSW — without worrying about potential conflicts.

On-Board Software

Integration of the MO model within an embedded software has proven difficult at the times, considering the extra resource overhead coming from the data model complexity. This was solved by implementing the MO communication stack from scratch in the OBSW, flattening its layers, and aiming to reduce the memory and computation overheads. Some functionalities of the MO have been opted out from the implementation, e.g., COM Archive, as discussed before.

NanoSat MO Framework

NMF as such has been built with short time to production in mind, which enforced large reuse of the existing MO implementation stack, developed in Java at the time. This has driven the technology choice for the entire framework, which is not very embedded-friendly. That in turn projects onto its overall resource consumption and adoptability on different systems. In OPS-SAT this issue was of no big concern, as the SEPP is running a relatively powerful computer based on Altera 5 SoC and 1 GB of RAM. This in turn provides performance at a similar level as a Raspberry Pi Single Board Computer, allowing it to run the Java Virtual Machine. However, due to a recent hardware degradation, the SEPP RAM allocation has been reduced to 512 MB by avoiding access to the HPS DDR RAM and effectively using the FPGA RAM instead, which is starting to prove difficult when attempting to run more complex applications [2].

The short-term mitigation of this involves memory profiling and optimisation of the MO Stack and the NMF. Longer term solution would aim at drastically reducing the NMF memory footprint by reimplementing its core components in a native technology like C or C++, such as with the NMF API Lite in C [1].

Light-Weight Mission Control System

ESA LWMCS is yet another technology validated largely during routine OPS-SAT operations, while being actively developed. The risk reduction approach involves agile development cycle, self-testing builds, CI/CD processes with multiple staging environments, and processing the user feedback on a sprint basis.

FMS/CFDP

With a highly experimental CFDP and FMS deployment topology involving using identical Java applications in space and on the ground, OPS-SAT is effectively validating such an approach in flight. This, combined with occasional TM/TC link dropouts or data corruption is sometimes proving difficult in operations.

The data systems team is actively mitigating the impact of various edge cases found in operations by reproducing them in a test environment and finding solutions to them. Often those improvements consist only of configuration changes to the CFDP daemons, thanks to the very high configurability of the ESA CFDP implementation.

FUTURE WORK AND DISCUSSION

MO services, NMF, CFDP FMS, the spacecraft, as well as the ground data systems all offer exciting potential for further development.

MO Services

The MO have shown great promise as the means to achieve interoperability between various space and ground systems. The authors propose to keep evolving the MO ecosystem and its reference implementations with the focus on:

- security-oriented features,
- supporting more implementation technologies and language bindings,
- efficient protocols and encodings aiming to reduce space link overhead,
- improving the persistence model,
- reducing resources footprint,
- certification of the code,
- reference monolithic stack for embedded systems.

NanoSat MO Framework

The NMF has been instrumental in OPS-SAT success by allowing quick turn-around of the mission applications, taking the advantage. The authors anticipate the need to not only maintain the current NMF stack, but also build the next version of it, following the same set of recommendations as stated in the previous point. According to the consultations with the industry, implementing NMF in a language that is lower level than Java, while keeping backwards compatibility with its current API in addition to exposing APIs in languages like Python, C++, Rust, would help its adoption across the growing New Space sector.

CFDP FMS

CFDP FMS daemon employs support for a Linux Filesystem in USErspace (FUSE), allowing to mount a remote filesystem transparently and queueing any I/O operations transparently to the user. It can be integrated in the OPS-SAT and future mission ground segments, providing high enough reliability of the underlying systems.

Considering the growing presence of high bandwidth and high availability links in the LEO, involving internet constellations and optical relay terminals, the future LEO missions might possibly rather employ a set of well-established Internet Protocol (IP) based solutions like SSH, FTP, rsync, leaving CFDP standard to the deployments in the higher orbits and deep space missions, where there is no IP-enabling infrastructure for the foreseeable future. It is also important to note that the CFDP could still be used over a lossy IP link to increase the reliability of the file transfers.

OPS-SAT

The authors are looking forward to the new possibilities for the OPS-SAT mission and its successors which keep opening thanks to the growing maturity and adoption of the new interoperability standards within the industry.

Ground Data Systems

OPS-SAT was demonstrated as the first flying satellite to be controlled by the new generation of the Mission Control System — the European Ground System Common Core (EGS-CC), meant to eventually replace SCOS-2000. Not only was this a great milestone marking the beginning of the operational adoption of the EGS-CC within the Agency, but also the ultimate test of the mission for its ability to push the envelope for the software across both ground and space [16, 17].

CONCLUSIONS

Using OPS-SAT as the technology demonstrator for MO Services and CFDP has provided an enormous feedback value to all teams responsible for the design, development, implementation, and operation of those standards. This turned out to be the perfect realization of the core mission principle — to allow rapid advancement of the technology readiness of the software via in-orbit validation and demonstration. OPS-SAT is paving the way for other missions and orbital platforms aiming to adopt those technologies soon, both within ESA and commercial endeavors. The employed solutions ease the mission integration with external users as well as decrease the general system complexity and the maintenance effort, thus increasing the mission return value.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the contribution of all members of the CCSDS working groups for Spacecraft Monitor and Control (MOIMS-SM&C) and CFDP Revisions (SIS-CFDPV1) in design, implementation, and testing of the standards. They would also like to thank the enormous contribution of the industrial consortium that made the entire project possible, especially Otto Koudelka, Cesar Coelho, Maximilian Henkel, Manuel Kubicka, and Reinhard Zeif at TU Graz.

REFERENCES

1. Evans, D., Labrèche, G., Mladenov, T., Marszk, D., Shiradhonkar, V., & Zelenevskiy, V. (2022). Agile Development and Rapid Prototyping in a Flying Mission with Open-Source Software Reuse On-Board the OPS-SAT Spacecraft. *AIAA SciTech Forum 2022*. https://doi.org/10.2514/6.2022-0648

2. Evans, D., Labrèche, G., Mladenov, T., Zelenevskiy, V., Marszk, D., & Shiradhonkar, V. (2022). OPS-SAT LEOP and Commissioning: Running a Nanosatellite Project in a Space Agency Context. *36th Annual Small Satellite Conference*.

3. Labrèche, G., Evans, D., Marszk, D., Mladenov, T., Shiradhonkar, V., Soto, T., & Zelenevskiy, V. (2022). OPS-SAT Spacecraft Autonomy with TensorFlow Lite, Unsupervised Learning, and Online Machine Learning. *2022 IEEE Aerospace Conference.*

4. Evans, D., Labrèche, G., Marszk, D., Bammens, S., Hernández-Cabronero, M.,

Zelenevskiy, V., Shiradhonkar, V., & Starcik, M. (2022). Implementing the New CCSDS Housekeeping Data Compression Standard 124.0-B-1 (based on POCKET+) on OPS-SAT-1. *36th Annual Small Satellite Conference*.

5. Kacker, S., Meredith, A., Cahoy, K., & Labrèche, G. (2022). Machine Learning Image Processing Algorithms onboard OPS-SAT. *36th Annual Small Satellite Conference*.

6. Mladenov, T., Evans, D., & Zelenevskiy, V. (2022). Implementation of a GNU Radio-Based Search and Rescue Receiver on ESA's OPS-SAT Space Lab. *IEEE Aerospace and Electronic Systems Magazine*, vol. 37, no. 5, pp. 4-12, 1 May 2022. https://www.doi.org/10.1109/MAES.2022.3143875

7. Segret, B., Bammens, S., Bras, S., Marszk, D., Shiradhonkar, V., Zelenevskiy, V., & Evans, D. (2022). On-board images to characterize a CubeSat's ADCS. *36th Annual Small Satellite Conference*.

8. Mission Operations Services Concept. CCSDS 520.0-G-3. Green Book. Issue 3. December 2010.

9. CCSDS MO Services Web Viewer accessed July 15, 2022, https://esa.github.io/mo.viewer.web/

10. Telemetry and telecommand packet utilization. ECSS-E-ST-70-41C. 15 April 2016.

11. CCSDS File Delivery Protocol – Implementers Guide. CCSDS 720.2-G-3. Green Book. Issue 3. April 2007.

12. Coelho, C. (2017). A Software Framework for Nanosatellites based on CCSDS Mission Operations Services with Reference Implementation for ESA's OPS-SAT Mission. PhD dissertation, Graz University of Technology.

13. Coelho, C., Koudelka, O., & Merri, M. (2017). NanoSat MO framework: When OBSW turns into apps. *2017 IEEE Aerospace Conference*, 1-8.

14. Coelho, C. B. W., Cooper, S., Koudelka, O. F. S., Merri, M., & Sarkarati, M. (2017). NanoSat MO Framework: Drill down your nanosatellite's platform using CCSDS Mission Operations services. In *Proc. International Astronautical Congress*.

15. CCSDS MO Services Web Viewer for NMF, accessed July 15, 2022, https://dmarszk.github.io/MOWebViewer4NMF/

16. First test of Europe's new space brain, accessed July 15, 2022, https://www.esa.int/Enabling_Support/Operations/Ground_Systems_Engineering/First_test_of_Europe_s_new_space_brain

17. Cubesat Low-Cost Control System (C2LOCO), accessed July 15, 2022. https://www.blog.visionspace.com/blog/c2loco