18th International Conference on Space Operations, Montreal, Canada, 26 - 30 May 2025.

SpaceOps-2025, ID # 281

Innovating ground software systems – a retrospective of OPS-SAT-1 mission

Dominik Marszk *[a], Daniel Fischer [a], Nuno Ramos Carvalho [a], Tim Oerther [b], Maximilian Henkel [c], Georges Labrèche [d]

[a] *European Space Agency, European Space Operations Centre, Darmstadt, Germany*mailto:
[b] *Terma GmbH, Darmstadt, Germany*
[c] *Graz University of Technology, Austria*
[d] *Tanagra Space, Queens, NY, USA*
mailto:
[*] *Corresponding Author dominik.marszk@esa.int*

Abstract

OPS-SAT-1 Flying Laboratory was a 3U CubeSat of ESA operated between 2019 and 2024. It served as an in-orbit demonstration and verification platform for a wide variety of projects from all over the world. 285 different experiments, submitted by 134 teams, were conducted during its lifetime. This software-defined satellite allowed to be quickly programmed for software and firmware experiments, execute them, and collect the results with a minimal turnaround of just one day.

Running platform and payload operations of this complexity with a constrained budget introduced a unique set of obstacles. This was further compounded by the unprecedented scale at which ESA exposed both ground and space systems to the outside world. Addressing these challenges required innovative solutions within the ground data systems. This paper outlines the characteristics of the mission with a focus on the software design drivers that made OPS-SAT-1 stand out from other ESA projects. It then elaborates on the solutions employed to address the discussed challenges, especially the adopted processes and developed software. Finally, a retrospective is presented on over 4 years of operations with an emphasis on how are the solutions developed for OPS-SAT-1 applicable to subsequent missions like OPS-SAT VOLT and OPS-SAT ORIOLE.

Keywords: DevOps, CubeSat, SmallSat, Satellite-as-a-service, Software testing, Fault tolerant systems, Reliability, Reconfigurable devices, Edge computing, Virtualization.

Acronyms/Abbreviations

ADCS	Attitude Determination and Control System
APID	Application Process Identifier
API	Application Programming Interface
CFDP	CCSDS File Delivery Protocol
CI/CD	Continuous Integration / Continuous Deployment
CLTU	Communications Link Transmission Unit
DRS	Data Relay Server
ECSS	European Cooperation for Space Standardization
EPS	Electrical Power System
FAPEC	Fully Adaptive Prediction Error Coder
FPGA	Field Programmable Gate Array
FMS	File Management Services
GNSS	Global Navigation Satellite System
I2C	Inter-Integrated Circuit
IPK	Itsy Package (Linux software package format)
LEOP	Launch and Early Orbit Phase
LWMCS	Lightweight Mission Control System
MAL	Message Abstraction Layer
MATIS	Mission Automation Toolkit for Integrated Systems
MC	Monitoring and Control (part of CCSDS MO)
MCS	Mission Control System
MCT	Mission Control Team
MIB	Mission Information Base
MICONYS	Mission Control System Software Suite (ESA)

MO	CCSDS Mission Operations (Services)
MUST	Mission Utility and Support Tools
NAK	Negative Acknowledgement
NMF	NanoSat MO Framework
OBSW	On-Board Software
OBC	On-Board Computer
ORIOLE	Optical Relay and Infrared Optics for LEO Experiments
PDU	Protocol Data Unit
PUS	Packet Utilization Standard
SaaS	Software as a Service
SCOS-2000	Spacecraft Control & Operations System (ESA MCS)
SDR	Software Defined Radio
SEPP	Satellite Experimental Processing Platform
VOLT	Versatile Optical Laboratory for Telecommunications

1. Introduction

1.1. Mission goals and novelty

OPS-SAT-1 was a 3U CubeSat mission launched by the European Space Agency (ESA) on 18 December 2019, designed as a dedicated in-orbit testbed for advanced software and operational technologies. Developed under ESA's General Support Technology Programme (GSTP), the mission aimed to validate innovative concepts under real flight conditions, providing a flexible and openly accessible platform for experimentation. In contrast to traditional ESA missions, OPS-SAT-1 prioritised rapid iteration, reconfigurability, and openness to external contributors, thereby serving as a technology demonstration pathfinder for future operational missions.

The mission's primary objective was to address the systemic barriers often encountered when introducing new technologies into flight systems, commonly referred to as the "has never flown, therefore will never fly" constraint. To that end, OPS-SAT-1 combined a capable experimental payload—the Satellite Experimental Processing Platform (SEPP) [1], based on a dual-core ARM processor with FPGA fabric—with a broad set of onboard sensors and actuators. This configuration enabled the execution of diverse software and firmware experiments, ranging from image processing and compression [2] to protocol validation, AI-based classification [3], and reconfigurable onboard control logic [4].

1.2. OPS-SAT-1 in ESA missions portfolio

OPS-SAT's operational concept was also novel within the ESA mission portfolio. It was the first ESA-operated mission to adopt an operations model driven entirely by software engineering concepts, with significant reuse of open-source tools and frameworks [5], major reliance on virtualised ground infrastructure, automated testing and deployment pipelines, community-facing interfaces such as the NanoSat Mission Operations Framework (NMF) and web-based mission control tools. These characteristics positioned OPS-SAT-1 not only as a technical demonstrator, but also as a platform for experimenting with new mission control paradigms and ground segment software architectures.

Within ESA's broader mission landscape, OPS-SAT-1 represented a deliberate shift toward cost-effective, rapid-cycle missions capable of supporting iterative development and experimentation. While not intended for operational service delivery or long-duration science return, OPS-SAT-1 complemented flagship science missions by accelerating the readiness and validation of technologies potentially applicable to them. Furthermore, it provided ESA's operations community with hands-on experience in managing a mission exposed to external users at an unprecedented scale, necessitating the development of robust safety and security mechanisms within the ground data systems and on the spacecraft.

This paper presents a retrospective analysis of the OPS-SAT-1 mission from the perspective of ground software systems. It highlights the software design drivers and engineering choices that enabled a high level of flexibility, discusses the architectural and procedural innovations introduced to support safe and efficient operations, and examines lessons learned over the course of more than four years of continuous activity. The final sections reflect on the applicability of the developed solutions in the context of upcoming missions such as OPS-SAT VOLT and OPS-SAT ORIOLE.

2. Mission Overview

2.1. Development and launch timeline

The OPS-SAT-1 mission originated as a proposal within ESA's General Study Programme (GSP) in 2011, with the objective of creating an in-orbit laboratory for testing novel software and firmware. A feasibility study was conducted in early 2012 using the ESA Concurrent Design Facility, and in 2013, an open call for experiment proposals

resulted in more than 100 submissions from ESA Member States. The space and launch segments were subsequently funded through the General Support Technology Programme (GSTP), while the development of the ground segment and operational concepts was carried out at ESA's European Space Operations Centre (ESOC).

The spacecraft was developed by a consortium led by TU Graz (technical prime), with industrial contributions from MAGNA STEYR, UNITEL (Austria), GomSpace (Denmark), MEW Aerospace, Berlin Space Technologies (Germany), GMV, and Space Research Centre (Poland). The final integration and testing were completed in 2019, followed by delivery to the launch provider. OPS-SAT-1 was launched aboard a Soyuz rocket on 18 December 2019 from Korou as part of Arianespace flight VS23, sharing the payload with other small satellites including CHEOPS, ANGELS, and EyeSat.

2.2. Key milestones

2.2.1. Launch and Early Orbit Phase (LEOP)

LEOP was initiated immediately following launch, but the mission encountered significant challenges from the outset [6]. No UHF packets were received during the initial passes, and subsequent attempts revealed degraded communications performance on this link. These issues were traced to a combination of onboard configuration problems and elevated noise levels at the receiving ground stations. Full two-way communication was not achieved until approximately two months post-launch, following configuration changes to the UHF transceiver and uplink power enhancements at the TU Graz (TUG) ground station.

2.2.2. Commissioning Phase

Full payload commissioning was delayed by degraded communications and some system-level instabilities such as the poor performance of the on-board I^2C bus [6]. Despite this, the team pioneered hybrid approaches using onboard software updates, onboard scripting, and file-based commanding to progressively bring the spacecraft to routine operations.

2.2.3. Mission Extension and Routine Operations:

Following the completion of commissioning in Q4 2020, OPS-SAT-1 entered routine operations. The mission was formally extended twice, with operational support continuing through 2024. During the entire 4.5 year period, the OPS-SAT-1 experimenter community spanned the globe, consisting of 134 teams and 285 unique experiments several of which transitioned into regular operational use (e.g., onboard image compression with a custom codec – FAPEC [7], autonomous AI-based imaging selection [3][8]). Routine operations were characterised by a high degree of automation, frequent software updates, and a mix of manned and unmanned ground passes.
OPS-SAT operations continued until the re-entry of the satellite on May 22, 2024.

2.3. Platform Architecture and Experimentation Capabilities

OPS-SAT-1's hardware architecture was divided into two functional domains: a minimalised conventional CubeSat bus and an experimental payload stack.

The bus components were provided by GomSpace and consisted of the following subsystems:

- **NanoMind A3200 On-Board Computer** for platform control and attitude determination.
- **UHF transceiver** for contingency and low-rate telemetry and telecommanding.
- **Electrical Power System (EPS)** including deployable solar panels and battery management.
- **GNSS receiver**.

The experimental stack was centred around the **Satellite Experimental Processing Platform (SEPP)**, developed by TU Graz. The SEPP is based on the Altera Cyclone V SX SoC, integrating a dual-core ARM Cortex-A9 Hard Processor System (HPS) with a reconfigurable FPGA fabric. The board operates under a custom embedded Linux distribution (Ångström, built via OpenEmbedded) and offers significant computing resources: 800 MHz CPU, 1 GB DDR3 RAM, and multiple peripheral interfaces.

Key payload and experimental components available to the SEPP system include:

- **S-band and X-band transceivers** (Syrlinks) with CCSDS TM/TC support and optional protocol bypass.
- **HD optical camera** (BST IMS-100) connected via high-speed data links.
- **Software-Defined Radio (LMS6002D)** enabling flexible RF experimentations.
- **iADCS-100** advanced attitude determination and control system, including miniature reaction wheels and a star tracker.
- **Optical receiver** for laser communication experiments.

The SEPP also hosted the **NanoSat Mission Operations Framework (NMF)**, a modular, service-oriented software stack based on CCSDS MO Services [9][8], enabling experimenters to build and deploy onboard applications in a structured and abstracted environment. This framework was critical in supporting the mission's core objectives of flexibility, safety, and remote access.

3. Ground Segment and Data System Architecture

The OPS-SAT-1 ground segment was uniquely architected to meet the dual requirement of supporting both traditional ESA Flight Control Team (FCT) operations and dynamic, externally-driven experiment workflows, integrating these responsibilities into a single Mission Control Team (MCT). This hybrid model required a fundamental departure from monolithic mission control paradigms, incorporating service-oriented interfaces, open-access layers, and robust safety mechanisms—together forming a blueprint for future software-defined space missions.

3.1. Small Mission Infrastructure Laboratory Environment (SMILE) Lab and mission control infrastructure

The mission operations were conducted from the SMILE Lab at ESA's European Space Operations Centre (ESOC), a dedicated control facility designed for flexible and agile small mission operations. The ground infrastructure was built atop the SCOS-2000 Mission Control System (MCS) and Mission Automation Toolkit (MATIS), reused and extended from larger ESA missions [10]. This was augmented by a suite of custom components developed specifically for OPS-SAT, including an MCS Data Proxy, MO Data Proxy, and various MO-based control tools and interfaces.

A key enabler of OPS-SAT's agility was the full virtualization and progressive containerisation of the monitoring and control environment, allowing rapid spin-up of new development and operational environments as needed. These instances were linked via a dedicated network backbone, isolated from the rest of ESA's infrastructure and separating operational, validation, and internet-facing domains, enabling safe parallel execution of flight and non-flight operations.

3.2. SCOS, CFDP and MO integration

OPS-SAT-1 was the first ESA mission to adopt the CCSDS Mission Operations (MO) Services [11] and the CCSDS File Delivery Protocol (CFDP) [12] as first-class elements of its operations stack [8]. These were deeply integrated into SCOS-2000 using a novel model-driven approach: MO interface descriptions were ingested nightly into a MIB generator pipeline, automatically producing SCOS-compliant command and telemetry definitions.

The CFDP implementation, configured in Class 1— Acknowledged Mode with immediate NAKs [12]— was deployed both in the ground segment and onboard (SEPP), enabling robust file-based uplink and downlink. This replaced many legacy TC/TM workflows with reliable, resumable file transfers. MO-based applications were similarly integrated into the ground system via the Data Proxy, ensuring seamless monitoring and control of the applications built upon the experimenter framework.

3.3. Experimenter access layers and data routing

OPS-SAT's architecture supported multiple concurrent users with varying access needs and technical expertise. To manage this, the ground segment exposed three key access layers:

- **Data Relay Server (DRS):** The main interface for experimenters to upload software and retrieve mission artifacts, also offering live TCP/IP stream of CCSDS Space Packets relayed onto and from the spacecraft. DRS was built around an SSH and SFTP server, protected by strong authentication and chroot isolation.
- **Ground MO Proxy:** A ground-side bridge implementing the MO protocol stack, enabling experimenters to interact with their onboard applications via standardized MO interfaces.
- **Lightweight Mission Control System (LWMCS):** A web-based MO-native tool provided to users for telemetry monitoring and telecommanding of their experiments.

Routing of telemetry and command data was managed by the **MCS Data Proxy**, a multiplexer and stream switcher capable of protocol conversion and APID-based filtering. This allowed isolation of experiment traffic and facilitated fine-grained control over which systems received which data flows.

3.4. Security domains and interface safety

The space-ground system was classified into five security domains: Operational network, Development and Validation Chain networks, Data Relay network, SEPP Execution Environment, and the Spacecraft Bus. Each domain was assessed for risk (based on threat likelihood and impact) and treated with tailored safeguards. [14]

Key security measures included:

- Strict isolation policies of the operational LAN from all external networks.
- Policy-enforcing, at the MCS Data Proxy level, of the data routed to and from the experimenter, together with keeping audit trails of the entire data throughout the entire mission.
- Automated logging and audit trails across all user-accessible systems.
- Mandatory validation of experiment binaries prior to flight execution.

- Auditable experiment packaging pipeline, executed by servers on ESA side.
- Controlled file transfer workflows using CFDP, and later SFTP, backed by access control policies.

This compartmentalized design ensured that external users could safely interact with the system while minimizing the risk of inadvertent or malicious interference with mission-critical operations.

4. The Experimenter-Centric Design

OPS-SAT-1 was conceived from the outset as a mission accessible, cost free, to a broad experimenter community—ranging from institutional stakeholders and academia to start-ups and individuals. This required rethinking the traditional approach to user operations. Instead of building a monolithic ground infrastructure gated by tight process control, OPS-SAT-1 exposed modular, sandboxed interfaces enabling safe and scalable user interaction with both the ground and space segments. The resulting ecosystem significantly reduced onboarding friction, allowing rapid prototyping and iteration directly on a live flight system.

4.1. Open access and onboarding

OPS-SAT-1 supported over 285 unique experiments during its lifetime, submitted by 134 teams with widely varying technical backgrounds. To accommodate this diversity, the mission adopted an **open-call experiment model**, supported by a lightweight onboarding workflow.

The onboarding process evolved to include:

- **Remote access to a test board** via scheduled time slots for experiment validation and debugging.
- **Testboards on demand**, the team had a set of development boards they could send out to experimenters to enable rapid prototyping especially for firmware experiments.
- **Documentation and training materials**, including API references, example applications, and interface definitions.
- **Pre-validated software and package skeletons**, provided as templates that bundled application logic, service declarations, and experiment metadata.

By introducing strong but user-transparent safety barriers (e.g., domain separation, file-based upload models, non-privileged execution environments), the OPS-SAT-1 team was able to keep low entry barriers while ensuring protection of the spacecraft from erroneous or malicious behaviour.

4.2. Live experimenter interfaces

OPS-SAT-1 provided experimenters with multiple interface layers, each supporting different use cases, programming styles, and abstraction levels. The architecture, illustrated at high level on Figure 1, enabled flexible access to both the onboard experiment applications and the underlying platform hardware, while preserving safety boundaries.

Experimenters could interact with their onboard applications using:

- **Direct Space Packet interfaces over TCP/IP -** Applications could expose custom, binary protocols by directly transmitting CCSDS space packets over encapsulated TCP links. This method offered the highest degree of control and performance but required detailed knowledge of the communication stack and ground routing configuration.
- **CCSDS MO Service interface –** Initially the default mechanism for experiment interaction was through the CCSDS Mission Operations (MO) Services, implemented via the NanoSat MO Framework (NMF). This abstracted the underlying communication, offering standard service definitions for commanding, telemetry, and file exchange. MO-based experiments could be monitored and commanded either programmatically or through dedicated user tools.
- **Lightweight MCS (LWMCS) -** Built atop the MO Services model, LWMCS offered a browser-based graphical interface for interacting with experiment applications. It automatically discovered available services and parameters, providing users with a plug-and-play dashboard for telecommanding, parameter visualisation, and log inspection.

In addition to the experiment application layer, OPS-SAT-1 exposed **platform-level APIs**, allowing advanced users to interact with onboard peripherals (e.g., camera, SDR, attitude sensors). These interfaces were available in two forms:

- **Native C/C++ APIs (payload drivers) -** Direct bindings to low-level driver libraries, intended for high-performance applications or experiments requiring full control over payload behaviour. These APIs interacted directly with the SEPP's device drivers and FPGA interfaces.

- **MO Platform Service wrappers -** A higher-level abstraction built into the NMF, exposing common device functions through standardised service calls. This layer offered a simplified, hardware-agnostic programming model and allowed cross-platform simulation using the same codebase. [13]

This tiered architecture enabled OPS-SAT-1 to serve both low-level developers and users seeking rapid prototyping through safe and abstracted interfaces. The system also supported parallel interaction with multiple experiment applications, thanks to the modularity of the MO service stack and flexible ground routing.

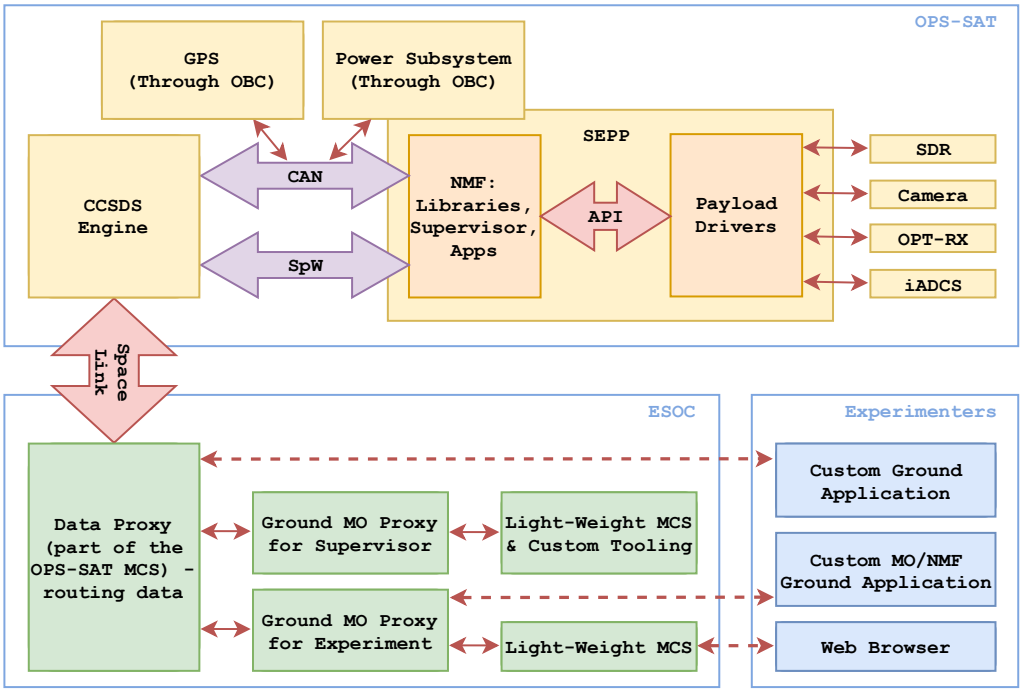

Figure 1: Overview of the OPS-SAT-1 live experimenter interfaces

4.3. Use of NMF SDK, and simulation tools

To streamline application development and validation, OPS-SAT-1 introduced two development environments for experimenters. Each experimenter could use either a containerised native toolchain for native platform interfaces access, or build Java-based NMF SDK including:

- The **NanoSat MO Framework (NMF) SDK**, allowed rapid creation of MO-compliant applications using a high-level service-oriented programming model.
- A simplified **simulator**, emulating payload interfaces and execution behaviour on desktop systems.
- Build tools for packaging experiments into IPK bundles, aligned with the onboard software distribution model.

These toolkits abstracted away the complexity of interacting with CCSDS-compliant infrastructure and enabled safe and consistent local development.

Together, these components formed a self-contained environment that was a simplification of the onboard software stack, significantly reducing the time required to go from concept to flight deployment.

5. Software-Defined Operations and DevOps Approach

OPS-SAT's operational concept was rooted in the principles of software-defined infrastructure: modularity, automation, and version-controlled configuration. Unlike conventional ESA missions where procedural operations dominate, OPS-SAT adopted DevOps practices throughout its lifecycle—from software development and deployment to routine commanding and anomaly resolution. This approach enabled the team to rapidly iterate, deploy updates safely, and scale operational support with limited personnel resources.

The ability to iterate and deploy software rapidly was further enabled by the decision to internalise onboard software (OBSW) development close to the mission launch, ensuring that key tools, frameworks, and payload control logic could be evolved in close coordination with the ground segment and automation infrastructure. This significantly reduced iteration time on each subsequent OBSW patch.

5.1. CI/CD pipelines and GitLab usage

Central to the mission's software lifecycle was a Continuous Integration and Continuous Deployment (CI/CD) pipelines based on **GitLab** and **Jenkins**. All onboard applications, ground tools, interface definitions (e.g., MO service XML), and configuration files were version-controlled and linked to automated build and validation processes.

Key features included:

- **Automated builds and unit tests** for both onboard and ground-side components.
- **Continuous integration and unit tests of mission interface definitions**, with automatic generation of SCOS-2000 MIBs and MO bindings.
- **Containerised build environments** to ensure consistency across development, validation, and production.
- **Branching strategies** that allowed experimental features to be tested on the Engineering Model before flight deployment.

This infrastructure drastically reduced the lead time between software development and in-orbit validation. Changes to the mission configuration or onboard software could be tested, reviewed, and deployed with full traceability and minimal operator effort.

5.2. Teleworking and remote operations adaptation

OPS-SAT operations were designed to support remote collaboration from the beginning, but the COVID-19 pandemic accelerated the need for a fully decentralised model. Thanks to the mission's software-defined infrastructure, the team was able to transition to **remote-only operations** with negligible impact on experiment cadence or system availability.

Adaptations included:

- **Remote access to the Engineering Model** for validation campaigns, using secured VPN access and remote desktop environments.
- **Internet and web-accessible dashboards and command interfaces**, such as telemetry on-line storage and plotting toolkit ESA Mission Utility and Support Tools (MUST), allowing engineers and experimenters to interact with the satellite from any location.
- **Email-delivered pass reports** with a link to ground station performance report, summary of performed activities and spacecraft status – an example shown on *Figure 2*
- **Heavy use of collaborative tools like MS Teams** reducing reliance on traditional shift handovers.

This operational flexibility ensured the mission remained fully functional through extended periods of teleworking, with core mission control responsibilities distributed across multiple countries.

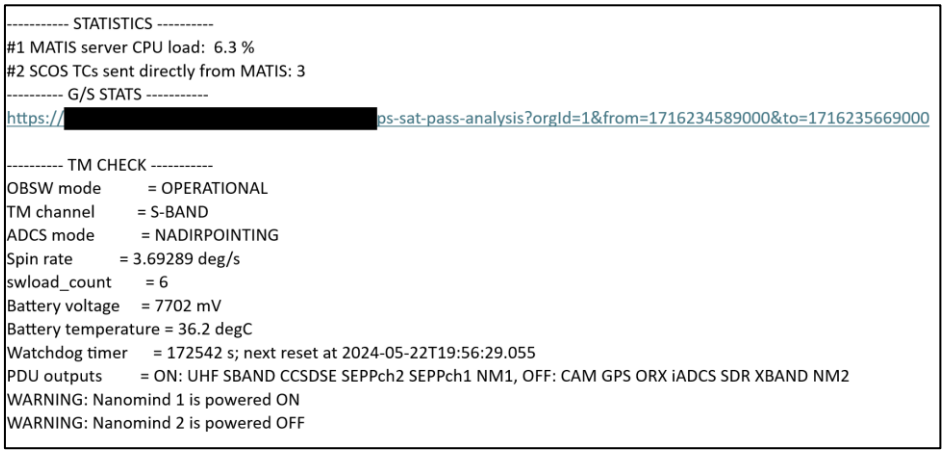

Figure 2 Example header of an S-Band pass report, for a pass over ESOC-1 ground station

5.3. Automation in daily operations

The mission was operated with a lean Mission Control Team, often without dedicated operators during nominal passes. This was made possible by a high degree of automation at all levels of the ground system:

- **MATIS (Mission Automation Toolkit for Integrated Systems) -** Used for scheduled commanding, health checks, and file-based data transfers. MATIS scripts could initiate CFDP uploads, restart onboard services, or execute recovery procedures without human intervention.
- **Experiment scheduling and data delivery pipelines -** Routine operations, such as experiment activations, log collection, and data downlink, were fully automated. Experimenters often received results within hours of uplink, without requiring operator interaction.

- **Automated watchdogs and status monitors -** Ground-side services monitored system health, uplink schedules, and spacecraft state-of-health parameters, triggering alerts or corrective actions when thresholds were breached.

The result was a mission architecture that achieved high availability and experiment throughput with minimal manual oversight - demonstrating that CubeSats, when equipped with software-defined infrastructure and modern DevOps practices, can achieve operational maturity on par with larger missions, even if their operational budget and subsequent staffing is usually much smaller.

6. In-Orbit Software Execution & Autonomy

OPS-SAT's defining feature was its ability to execute complex and adaptive software directly onboard the spacecraft. The Satellite Experimental Processing Platform (SEPP), equipped with a dual-core ARM processor and FPGA fabric, provided a flexible and powerful environment for deploying experiment applications, running automation scripts, and validating onboard autonomy concepts. This section summarises the practical mechanisms, frameworks, and achievements in enabling in-orbit software innovation.

6.1. Operational tools on SEPP (Python scripts, OBSW updates)

SEPP supported in-flight execution of user-defined scripts and payload logic updates via a sandboxed Linux environment. Routine operations frequently relied on Python-based utilities running directly on SEPP to execute image acquisition, file transfer orchestration, or experiment lifecycle management. This scripting interface was used not only by experimenters but also by the core mission team for system-level tasks and platform introspection.

Onboard software updates—including binary patches to the mission control software—were regularly uplinked and deployed via CFDP and MATIS automation. These updates were validated in the Engineering Model before flight and, in critical cases, delivered as delta-patched images to minimise bandwidth consumption.

The SEPP environment also supported a virtual file system for configuration and logging, enabling file-based command workflows, experiment self-scheduling, and persistent state tracking—all without involvement of the platform OBC or traditional telecommanding.

6.2. Use of open-source frameworks (e.g., TensorFlow for SmartCam)

OPS-SAT successfully demonstrated the feasibility of deploying modern open-source frameworks in orbit. One highlight was the SmartCam experiment, which used a TensorFlow Lite model to perform onboard image classification and scene selection. The SmartCam application was trained on the ground using open datasets, deployed via the application packaging process, and executed autonomously onboard. [3]

In addition to AI workloads, SEPP hosted multiple experiments using open-source libraries for image compression, signal processing (including GNU Radio framework [15] [16]), and protocol analysis. This validated the ability to operate computationally intensive, community-developed software stacks within a constrained CubeSat environment.

The success of these experiments demonstrated not only the technical feasibility, but also the benefits of reusing and validating terrestrial software frameworks in space—shortening development cycles and lowering the barrier to entry for contributors.

6.3. Onboard autonomy and AI applications

Several experiments explored **onboard autonomy**, leveraging the SEPP's processing power and data access flexibility. Examples include:

- **Autonomous imaging decisions** based on onboard classification inference, allowing the spacecraft to prioritise downlink of relevant data and discard irrelevant scenes.
- **Closed-loop experiment scheduling**, where experiment activation was triggered by sensor data or orbital parameters (e.g., lighting conditions or target visibility).
- **Adaptive system health management**, using lightweight scripts and parameter monitoring to detect anomalies and self-correct (e.g., by restarting failing applications).

Importantly, many of these autonomous behaviours were integrated using the **NanoSat MO Framework (NMF)**, which provided a structured environment for telemetry monitoring, parameter evaluation, and rule-based action execution. The platform's modularity allowed combining autonomy logic with system-level protections, ensuring that experimental autonomy could not compromise platform integrity.

OPS-SAT thus demonstrated that low-cost nanosatellites, when equipped with the right software abstractions and execution platform, can support a wide range of autonomous concepts—bridging the gap between software experimentation and operational capability.

7. Lessons Learned from Over 4 Years of Operations

7.1. Commissioning through experimentation

Traditional commissioning workflows proved impractical under OPS-SAT's communication constraints. The flexibility of the SEPP platform and its compatibility with scripting, remote file transfers (via CFDP), and onboard autonomy enabled an alternative path: treating commissioning itself as a series of experiments. This shift allowed for iterative reconfiguration and diagnostics even during poor link conditions, effectively validating that high-risk, software-centric platforms can be commissioned incrementally through remote experimentation.

7.2. Scaling and maintaining mission-critical ground software

OPS-SAT-1 introduced multiple layers of ground software: SCOS-2000, Mission Automation (MATIS), LWMCS, MCS Data Proxy, Ground MO Proxy, and more. With 134 active experimenter teams and evolving experimenter needs, the team had to implement robust CI/CD pipelines and nightly integration tests. The modular design of the data systems allowed for isolated evolution of ground-side components without impacting ongoing flight operations. Lessons from this architecture have already informed ESA's future ground systems like EGOS-MG.

7.3. Reliability trade-offs in rapid prototyping

OPS-SAT-1 deliberately embraced software and hardware elements that were still maturing. This resulted in several in-orbit anomalies: partial RAM failures, intermittent communication links, and thermal-induced hardware instabilities. However, fallback mechanisms, dual-core redundancy, and the capability to patch, reconfigure, or replace subsystems in software ensured mission continuity. A key takeaway is that platform resilience in software-defined missions must be achieved through reconfigurability and autonomous fault recovery, rather than rigid hardware reliability.

7.4. Experimenter development and operations support

OPS-SAT-1 supported a wide range of users—from universities to commercial developers—with diverse levels of experience. The onboarding process, initially resource-intensive, was streamlined through some prebuilt toolkits, remote access to a test board, templated IPK packaging, and clear validation protocols. The team also established a tiered support model, balancing direct engagement with automation and documentation. Key innovations included automated experiment telemetry routing, real-time dashboards, and sandboxed interface layers to isolate user activities from critical systems.

Typical experiments could either be native applications, or Java-based NMF applications, with vastly different build and packaging processes, providing heterogeneous and sometimes complex experience across the experiments landscape. For similar missions in the future, a more streamlined and containerised environment for development, validation, and operations of user applications would provide the best value.

8. Impact on Future Missions

The OPS-SAT-1 mission provided an end-to-end validation of software-defined operations, modular ground infrastructure, and open experimenter access to a live ESA spacecraft. Its success demonstrated that rapid development cycles, agile tooling, and community-facing interfaces are not only viable but beneficial in operational contexts. These insights have already informed follow-up missions and are shaping the evolution of ESA's ground system architecture.

8.1. OPS-SAT-2 missions: OPS-SAT VOLT and ORIOLE

OPS-SAT-1's architectural concepts and operational model directly influenced the design of its successors, currently in preparation:

- **OPS-SAT VOLT -** Versatile Optical Laboratory for Telecommunications **-** builds upon OPS-SAT-2 mission concept designed using Concurrent Design Facility (CDF) process, introducing optical communication and quantum key distribution capabilities onto the platform. The mission will continue the software validation model, with a focus on high-performance onboard computing, including next-generation FPGA acceleration, AI/ML workloads, and advanced file delivery pipelines.
- **OPS-SAT ORIOLE** - Optical Relay and Infrared Optics for LEO Experiments - is also based on OPS-SAT-2 mission concept, with particular focus on high-bandwidth optical transmission and relaying capabilities.

Both follow-ups intend to preserve OPS-SAT-1's experimenter-centric philosophy while expanding hardware and software capabilities offered by the platform.

8.2. Reuse of ground segment architecture and concepts

OPS-SAT's ground segment was not only mission-enabling but also forward-looking, serving as a prototype for modern, scalable infrastructure applicable to missions of similar profile. Its core architecture and associated processes will be directly reused or evolved for future missions.

Key reuses include:

- **SMILE Ground Segment Infrastructure -** The SMILE ground segment and ground stations are being reused as the environment supporting exploitation of future experimentation platforms like CyberCube, OPS-SAT VOLT, ORIOLE.
  A major enhancement underway is the integration of an optical ground station interface into the SMILE segment, expanding data return capabilities via high-rate optical downlinks—an evolution designed to match the needs of VOLT and ORIOLE.
- **Development and Operational Processes -** The mission's GitLab-based CI/CD infrastructure, machine-readable, version-controlled interface definitions, and virtualised testing environments have been retained for subsequent missions. These DevOps-aligned workflows have demonstrated their ability to lower operational cost and reduce latency between development, validation and flight.
- **Evolved Experimentation and Verification Framework -** The experiment onboarding and validation framework, built around Dockerised development environments, formalised interface contracts, and controlled FlatSat access, will be adapted and expanded for new missions.

This reuse of infrastructure and processes not only reduces the ramp-up time for new missions but also ensures continuity of operational philosophy.

8.3. Applying OPS-SAT-1 lessons to operational flexibility

OPS-SAT-1 offered a unique window into how operational agility can be achieved in institutional space missions. Lessons that have cross-mission relevance include:

- **Separation of concerns in ground software design -** Domain-based compartmentalisation allowed for tight safety controls while enabling user access to selected subsystems—a model now influencing multi-tenant architectures.
- **Embracing failure-tolerant design for experimentation -** OPS-SAT treated recoverability as a design goal and a safe-mode as a regular mission state. SEPP resets, watchdogs, and redundant pathways ensured that experiment failures did not escalate into mission-impacting events.
- **Flexible operations staffing models -** The heavy use of automated passes, relying on scheduled interventions, and remote collaborative tooling proved to be effective for constrained presence on-site and small staffing.

OPS-SAT's legacy lies in proving that sophisticated mission software operations can be decentralised, modular, and open, without compromising safety or reliability. As ESA expands its portfolio of fast-track, low-cost missions, the patterns established by OPS-SAT are serving as architectural and procedural templates.

9. Conclusion

9.1. Summary of innovations and challenges

OPS-SAT-1 marked a transformative step in ESA's approach to mission operations, becoming the first mission to truly implement the concept of a **software-defined, reprogrammable spacecraft** accessible to external users. Its success came not from a single breakthrough, but from the cumulative impact of multiple operational and architectural firsts:

- **First ESA mission to operate entirely using CCSDS MO Services** for onboard and ground system interaction.
- **First operational use of CFDP** for routine, file-based software upload and telemetry return.
- **First ESA mission to use regular TCP/IP and SSH connection over a space link** to the payload computer for operating the spacecraft, retrieving telemetry and maintenance tasks.
- **First spacecraft to offer in-orbit deployment of user-developed applications**, with real-time access via standardised service interfaces.
- **First ESA mission to support containerised CI/CD pipelines**, enabling safe and rapid experiment integration and automated uplink workflows.
- **First in-orbit use of deep neural networks**, with SmartCam demonstrating onboard data filtering and autonomous decision-making.
- **First real-time wider user access to an ESA spacecraft via browser-based tools** like Grafana, MUST, LWMCS.
- **First ESA mission operated under a distributed, teleworking-friendly model** throughout its routine operations phase—enabled by automation, remote access, and virtualised infrastructure.

These innovations enabled OPS-SAT to support 285 unique experiments submitted by 134 teams, ranging from novel protocols and codecs to in-orbit reconfiguration of FPGAs and embedded platforms. The mission achieved a **high cadence of experiment turnarounds**, often validating and executing new payload software within days.

However, these advancements also introduced novel operational challenges. Issues such as hardware degradation (e.g., partial RAM failure, flash memory failure due to wear and radiation [17]), noisy RF environments, and experiment-induced faults required robust fallback mechanisms and a shift in mindset—from failure prevention to **resilience through reconfigurability** [6]. The success of OPS-SAT lay in its ability to absorb failures, recover autonomously, and maintain continuity of service even in degraded configurations.

By validating the core tenets of modular, service-based operations, OPS-SAT laid the foundation for next-generation missions—both institutional and commercial. It proved that **small satellites can serve as serious platforms for software innovation**, enabling secure, dynamic, and scalable operations that reflect the needs of a more agile space sector.

9.2. OPS-SAT's role in shaping future software-defined missions

OPS-SAT-1 has demonstrated that a small satellite can serve as a capable, multi-user, software-defined in-orbit laboratory, supporting real-time experimentation through virtualised ground infrastructure, open APIs, and safe user isolation mechanisms. These concepts have now found fertile ground beyond institutional missions, influencing the development of both new ESA CubeSat missions and commercial ventures.

The OPS-SAT software stack—especially the experimenter framework, the experiment validation chain, and the concept of a modular software-defined payload—has shaped the thinking in follow-on projects such as **OPS-SAT VOLT** and **OPS-SAT ORIOLE**. These missions are building on the SMILE ground infrastructure, experimenter access layers, and deployment strategies pioneered in OPS-SAT-1.

In parallel, a growing number of commercial satellite missions are embracing similar paradigms. Companies now offer access to small, software-defined satellites as a service, where users can deploy and run code in space via standardised APIs, often built atop commercial DevOps platforms. Examples include satellite-as-a-service providers such as Open Cosmos or EnduroSat [18][19], who offer programmable payload hosting and cloud-integrated mission operations interfaces. Many of these services share the same goals of lowering barriers for in-orbit validation, albeit with commercial constraints around cost, access, and security.

In recognition of its pioneering approach to software-defined operations, open experimentation, and mission agility, the OPS-SAT-1 team was awarded the SpaceOps Award for Outstanding Achievement in 2023 [20]. This honour reflects the collective innovation and dedication of all contributors to the mission and affirms OPS-SAT-1's role in shaping the future of agile space operations.

Acknowledgements

The authors would like to acknowledge the contributions of former team members whose work laid the foundation for the OPS-SAT-1 mission's success. We also extend our sincere thanks to the experimenter community, whose engagement, creativity, and persistence were instrumental in demonstrating the mission's full capabilities.

References

[1] R. Zeif, M. Henkel, A. Hörmer, M. Kubicka, M. Wenger, O. Koudelka, The redundancy and fail-safe concept of the OPSSAT payload processing platform, Proceedings of the 69th International Astronautical Congress, 2018.

[2] S. Kacker, A. Meredith, K. Cahoy, G. Labrèche, Machine Learning Image Processing Algorithms Onboard OPS-SAT, Proceedings of the 36th Annual Small Satellite Conference, Logan, UT, USA, Aug. 2022.

[3] G. Labrèche, T. Mladenov, D. Evans, OPS-SAT Spacecraft Autonomy with TensorFlow Lite, Unsupervised Learning, and Online Machine Learning, Proceedings of the 2022 IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT), Pasadena, CA, USA, Sep. 2022.

[4] L. Chavier et al., Deploying Artificial Intelligence Capabilities by Hybridizing a Neural Network on a Satellite, Proceedings of the AIAA ASCEND Conference, Las Vegas, NV, USA, Oct. 2023.

[5] G. Labrèche, T. Mladenov, Open-Source Software in Space Operations, Space Education and Strategic Applications, vol. 1, no. 2, pp. 1-12, Dec. 2021.

[6] D. Evans, G. Labrèche, T. Mladenov, D. Marszk, V. Zelenevskiy, V. Shiradhonkar, OPS-SAT LEOP and Commissioning: Running a Nanosatellite Project in a Space Agency Context, Proceedings of the 36th Annual Small Satellite Conference, Logan, UT, USA, Aug. 2022.

[7] J. Portell, A. Martí, R. Iudica, D. Evans, V. Zelenevskiy, Image and Radio-Frequency Data Compression for OPS-SAT Using FAPEC, presented at the On-Board Payload Data Compression Workshop (OBPDC), Noordwijk, The Netherlands, Oct. 2022.

[8] D. Marszk, D. Evans, T. Mladenov, G. Labrèche, V. Zelenevskiy, V. Shiradhonkar, MO Services and CFDP in Action on OPS-SAT, Proceedings of the 36th Annual Small Satellite Conference, Logan, UT, USA, Aug. 2022.

[9] C. Coelho, O. Koudelka, M. Merri, NanoSat MO Framework: When OBSW Turns Into Apps, IEEE Aerospace Conference, 2017.

[10] ESA, MICONYS Software Suite, https://esoc.esa.int/services-software (accessed 31.03.25).

[11] CCSDS Mission Operations Services Concept – Green Book, CCSDS 520.0-G-3, December 2010.

[12] CCSDS File Delivery Protocol – Implementers Guide, CCSDS 720.2-G-3, April 2007.

[13] CCSDS MO Services Web Viewer for NMF, https://esa.github.io/mo.viewer.web/?b=nmf_dev (accessed 31.03.25).

[14] D. Marszk, J.L. Feiteirinha, B. Fischer, D. Taubert, T. Graber, A. Lofaldli, M. Sarkarati, D. Evans, M. Merri, OPS-SAT – Opening a Satellite to the Internet, 10th IAA Symposium on Small Satellites for Earth Observation, Berlin, Germany, 2019.

[15] T. Mladenov, D. Evans, V. Zelenevskiy, 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.

[16] T. Mladenov, G. Labrèche, T. Syndercombe, D. Evans, Augmenting Digital Signal Processing with Machine Learning Techniques Using the Software Defined Radio on the OPS-SAT Space Lab, Proceedings of the 73rd International Astronautical Congress, 2022.

[17] M. Henkel et al., "Mitigating and Recovering from Radiation Induced Faults in Non-Hardened Spacecraft Flash Memory," 2024 IEEE Aerospace Conference, Big Sky, MT, USA, 2024.

[18] Open Cosmos, Open Constellation – The Shared Earth Observation Infrastructure, https://www.open-cosmos.com/open-constellation (accessed 31.03.25).

[19] EnduroSat, Shared Satellite Service, https://www.endurosat.com/wp-content/uploads/2021/08/EnduroSat_Shared_Satellite_Service_Presentation_08_2021.pdf (accessed 31.03.25).

[20] SpaceOps, SpaceOps Award Past Recipients, https://www.spaceops.org/awards/past-recipients/ (accessed 31.03.25)