In-Orbit Experimental Validation of Autonomous Operations within the AIX Satellite-as-a-Service Framework

Christian Cardenio
AIKO
Torino, Italy
christian@aikospace.com

Riccardo Maderna
AIKO
Torino, Italy
riccardo@aikospace.com

Romeo Casesa
AIKO
Torino, Italy
romeo.casesa@aikospace.com

Margo Morgese
AIKO
Torino, Italy
marco.morgese@aikospace.com

Johan Lennart Westin
AIKO
Torino, Italy
johan.westin@aikospace.com

Nadir Casciola
AIKO
Torino, Italy
nadir.casciola@aikospace.com

***Abstract*—As Low Earth Orbit (LEO) constellations continue to scale, traditional ground-centric operational paradigms struggle to meet increasing demands in latency, flexibility, and operational cost. The AIX (AI-eXpress) mission series addresses these limitations by introducing a service-oriented satellite architecture that enables dynamic in-orbit resource usage and application deployment. This paper presents the in-orbit experimental activities of orbital_OLIVER, an autonomous operations software suite currently flying onboard the AIX mission, developed in collaboration with Planetek Italia and D-Orbit.**

orbital_OLIVER provides goal-driven planning and onboard decision-making functions, operating as an application layer on top of the AIX service framework. Flight experiments target autonomous mission execution features including application-level fault handling, controlled restart, operational mode switching, and dynamic reconfiguration following mission knowledge updates. These capabilities are assessed on operational hardware to evaluate robustness, controllability, and integration with the AIX service lifecycle. Within the Satellite-as-a-Service paradigm enabled by AIX, such experiments aim to reduce operational coupling with the ground segment while supporting flexible Earth Observation services and multi-tenant usage scenarios. The results contribute to the validation of onboard autonomy as a practical enabler for scalable and responsive LEO service platforms.

I. Introduction

In traditional Earth Observation (EO) missions, timelines are typically generated on the ground, simplifying system maintenance and operational control. However, this approach inherently introduces latency and limits the ability to respond promptly to events detected onboard. Evidence from the FireBird/VAMOS mission has shown that, in a ground-centric architecture, the response to an onboard-detected event necessarily spans at least two subsequent communication windows. At the same time, these studies highlighted how hybrid on-board/on-ground approaches can significantly reduce this delay without transferring the full operational complexity to the spacecraft [1].

In recent years, research in this area has evolved along three main directions.
The first focuses on autonomous planning and scheduling of observations. Early implementations in EO missions such as FireBird have been extended by more recent work on intelligent planning for distributed satellite systems, as well as decentralised replanning strategies for event-driven observation scenarios [1].
The second direction is dynamic targeting, where look-ahead data are analysed in near real time to guide subsequent observations within the same orbital pass. A recent study by NASA's Jet Propulsion Laboratory shows that, for orbits at approximately 500 km altitude, a look-ahead angle of 45–50 degrees results in operational windows on the order of 30–60 seconds [2]. Under these constraints, edge intelligence is no longer optional but becomes a fundamental architectural requirement .
The third direction concerns data-driven automation of telemetry monitoring. Results from OPS-SAT experiments indicate that anomaly detection on real telemetry remains challenging due to signal fragmentation, missing data, and the limited availability of benchmark datasets. At the same time, the emergence of public datasets and reproducible evaluation baselines is beginning to enable more rigorous and objective assessment of model performance. The OPSSAT-AD dataset, released in 2025, was specifically designed to address this gap, while earlier studies on OPS-SAT telemetry monitoring had already demonstrated accuracies of up to 98.4% on unseen test data in targeted supervised configurations [3].

Within this landscape, AI-express (AIX) [4] occupies a distinctive position. It is not simply an onboard autonomy demonstrator, nor just an edge processing platform. Rather,

it represents a service-oriented infrastructure that integrates EO payloads, an abstract software environment, resource orchestration, and an "app store" / "satellite-as-a-service" model. Publicly available sources from ESA and Planetek describe AIX as a testbed for AI, blockchain technologies, in-orbit resource servitisation, and user-configurable workflows [5]. This shifts the focus from the autonomy of individual algorithms to the autonomy of entire application workflows.

This shift constitutes the key differentiating contribution of AIX compared to more narrowly focused demonstrations of planning, cloud avoidance, or anomaly detection.

II. Mission Context and Role of orbital_OLIVER within AIX

The AI-eXpress mission series implements a service-oriented in-orbit architecture in which onboard sensing, edge processing, application deployment, resource management, and ground-service interaction are treated as parts of a single operational chain [5]–[7]. AIX is not conceived as a standalone payload experiment, but as an integrated Satellite-as-a-Service environment combining a hosted space segment, a ground-based service backend, an application access layer, and an onboard execution framework. In this architecture, the spacecraft provides the physical hosting and platform services, while AIX provides the software-defined service layer through which onboard resources, Earth Observation data, processing functions, and application workflows can be exposed, configured, validated, and operated.

The AIX product and service model is structured around three main enabling elements. The AIX High-Performance Computing Platform (HPCP or AIX-P1) provides the high-performance onboard computing capability required for AI/ML execution and secure resource accounting. The AIX Software Framework (AIXF or AIX-P2) extends this capability into an onboard software framework able to host applications, orchestrate runtime services, and interface with spacecraft avionics and payloads. The AIX Software Development Kit (IOADK or AIX-P3) provides the development and application packaging environment used to prepare workflows and applications for execution on AIX-P2-enabled assets. On top of this product baseline, AIX-S1 exposes onboard services to hosted third-party payloads, while AIX-S2 enables application-driven Earth Observation and analytics services through an AppStore-like access model. The result is a distributed space-ground service architecture in which user requests, onboard execution, resource allocation, telemetry, data products, and service-level constraints are managed as a coherent lifecycle rather than as isolated mission operations.

Within this framework, orbital_OLIVER provides the autonomous operations layer [8]. Its role is to transform pre-approved mission objectives, operational constraints, telemetry, payload state, application status, and execution feedback into admissible onboard decisions. orbital_OLIVER is integrated within the AIX payload software stack as the autonomy extension associated with onboard planning and execution. It does not replace the deterministic execution mechanisms of the spacecraft or the OBCP-based command infrastructure; instead, it supervises the active operational context, monitors task execution and relevant events, and updates the task sequence when local conditions require adaptation. This positioning preserves the predictability required by spacecraft operations while enabling the responsiveness expected from a service-oriented orbital platform.

The AIX planning concept is organised across three complementary layers. Long-Term Planning defines strategic allocation of resources and baseline mission constraints over extended planning horizons. Short-Term Planning translates this strategic baseline into executable activities including acquisition opportunities and customer-specific execution windows. orbital_OLIVER then operates inside the approved planning envelope, providing the onboard layer that handles local execution monitoring, priority-based arbitration, operational mode switching, controlled recovery, and bounded replanning. This layered approach is central to the validation strategy of the mission: autonomy is not evaluated as an unconstrained decision-maker, but as a controlled software function operating inside mission-certified temporal, resource, and safety boundaries.

For the purposes of the in-orbit validation campaign, orbital_OLIVER is therefore assessed as part of the end-to-end AIX service lifecycle. The relevant behaviours include application-level fault handling, controlled restart, module recovery, reconfiguration, operational mode switching, payload and platform interface management, output interface management, and acquisition-pipeline orchestration. In the acquisition scenario, orbital_OLIVER commands image acquisition, triggers the selected image analysis application, receives the processing response, and uses that response to decide whether the acquired image shall be retained or discarded. This directly connects onboard autonomy with application execution and service-level data handling, which is the operational pattern required by future dynamic Earth Observation and multi-tenant service scenarios.

The flight campaign is organised as an incremental IOD path. AIX-1p acts as the precursor step, focused on the early consolidation of the service-oriented execution concept and the core software services required by the AIX operational model. AIX-1 then provides the first flight implementation of the AIX payload on the ION platform, supporting validation of the computing node, onboard software execution, low-latency service elements, and the initial integration of AIX functions with operational mission procedures. AIX-1+ extends the configuration with a more capable reference flight baseline, enabling broader validation of the service architecture, payload interfaces, and advanced autonomy functions under a richer operational context. This staged approach allows orbital_OLIVER to be introduced and verified progressively: first as a controlled software function, then as an integrated onboard supervisor, and finally as an operational autonomy layer supporting the Satellite-as-a-Service execution model [9].

This progression is important because it frames the val-

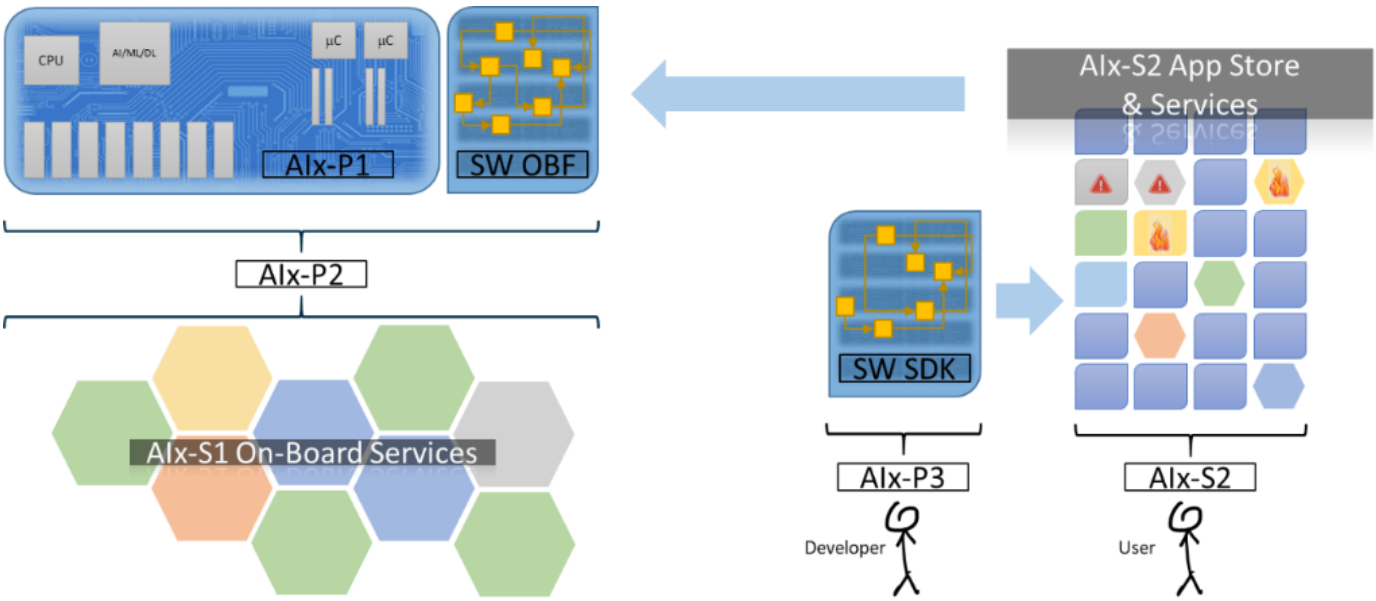

Fig. 1. AIX Product Service Model

idation of orbital_OLIVER in the correct system context. The objective is not only to demonstrate that an autonomy application can run onboard, but to show that application-level autonomy can be integrated, verified, and operated within a real service framework where user requests, payload resources, onboard processing, low-latency delivery, mission planning, and safety constraints are all part of the same operational chain.

III. Software Architecture of orbital_OLIVER in the AIX stack

At software level, orbital_OLIVER is implemented as a modular autonomy engine operating through a continuous onboard decision loop. The loop can be described as a sequence of five recurring functions: collect, evaluate, replan, execute, and report.

In the collect phase, orbital_OLIVER ingests platform telemetry, payload status, user-defined goals, anomaly-related information, and mission context. In the evaluation phase, it assesses plan feasibility, resource availability, timing, visibility, conflicts, and expected impact on mission return. In the replan phase, it updates scheduled operations within the authorised planning envelope. In the execute phase, it applies selected actions under predefined mission-safe rules. Finally, in the report phase, it synchronises the resulting state with the ground segment without interrupting onboard continuity.

This loop-based formulation is particularly suited to AIX because the mission is built around configurable onboard services rather than a fixed single-purpose payload chain. AIX enables users to configure acquisition and processing workflows, select or upload applications, and deploy services on demand as part of a Satellite-as-a-Service model [10]. orbital_OLIVER therefore operates as the autonomy layer that keeps this configurable execution model coherent at runtime: it supervises active tasks, reacts to relevant events, and adapts the operational sequence while preserving compatibility with the broader service framework.

A second key design choice is the separation between the reusable autonomy core and the mission knowledge layer. The reusable core implements the generic mechanisms for state ingestion, task reasoning, scheduling, execution supervision, and reporting. The mission knowledge layer captures AIX-specific elements: available services, operational modes, payload interfaces, resource envelopes, application constraints, service priorities, and allowable reconfiguration paths. This separation allows the same autonomy software principles to be reused across missions while constraining each deployment through explicit, mission-approved knowledge. It also makes controlled restart, operational mode switching, and dynamic reconfiguration meaningful verification targets, because these behaviours depend on the correct interaction between generic reasoning logic and AIX-specific operational knowledge.

The software architecture is designed to integrate into existing mission software rather than replace it. orbital_OLIVER is positioned as an application-level autonomy capability that interfaces with onboard execution services, payload-side processing applications, telemetry sources, and mission operations tools. In the AIX stack, deterministic platform functions and command execution remain governed by the existing spacecraft and OBCP mechanisms, while orbital_OLIVER provides bounded decision-making above them. This preserves a clear authority boundary: the autonomy layer may update application-level sequencing and operational context, but only within pre-defined safety, resource, and mission-planning constraints.

This authority boundary is central to the verification strategy. In conventional software verification, correctness can often be assessed by checking whether a component produces

the expected output for a given input. In AIX, the validation problem is broader: orbital_OLIVER must demonstrate that it can make admissible decisions while remaining observable, controllable, and composable inside a hosted multi-application environment. This includes verifying that decisions can be traced back to input telemetry, mission goals, resource constraints, and configuration knowledge; that replanning actions remain inside the authorised planning envelope; and that the resulting state can be reported back to the ground segment for reconciliation with mission operations.

The AIX application model further increases the importance of this architecture. The platform is designed to support resident applications, uploaded workflows, and containerised or graph-based processing chains. As a result, orbital_OLIVER must not be validated only as a planning component. It must also be validated as a supervisory software layer interacting with application lifecycle mechanisms: application activation, configuration update, execution monitoring, output handling, and reconfiguration after mission knowledge updates. This is why the V&V campaign treats OLIVER behaviours such as module restart, mode switching, interface management, and acquisition-pipeline orchestration as central test targets rather than as secondary robustness checks.

From a software engineering standpoint, this approach is consistent with the need to treat onboard autonomy as product software within the broader space system lifecycle. ECSS-E-ST-40C [11] frames space software engineering as covering requirements definition, design, production, verification and validation, transfer, operations, and maintenance across space and ground segments. For AIX, this means that orbital_OLIVER is verified not only at unit or component level, but through its operational interfaces and its role in the end-to-end service chain. Where packetised telemetry, telecommand, and data exchange patterns are involved, the architecture remains compatible with CCSDS-style space data handling principles, including standard packet-based services and interface discipline [12].

The resulting software architecture provides the basis for the validation campaign described in the following sections. The key question is not simply whether orbital_OLIVER can execute onboard. The relevant question is whether it can support autonomous application-level operations in a service-oriented mission, while preserving safety boundaries, operational traceability, ground synchronisation, and compatibility with configurable onboard applications.

IV. Verification and Validation Approach

The verification and validation strategy for orbital_OLIVER is structured as a progressive campaign spanning software verification, interface validation, system-level testing on representative hardware, and in-orbit experimentation. This approach is reflected within the testing process, shown in Figure 2 and in the role of orbital_OLIVER within AIX: the software is not validated only as an autonomous planning component, but as an operational function embedded in a service-oriented mission lifecycle. AIX itself is designed as an in-orbit testbed for low-latency, AI-enabled, service-based operations, where onboard resources and workflows can be configured and used on demand [5]. The V&V campaign therefore focuses on the ability of orbital_OLIVER to operate autonomously while remaining bounded, observable, and compatible with mission supervision from ground.

Four validation layers are foreseen.

The first validation layer addresses software correctness in isolation. At this level, the autonomy loop is exercised under controlled input conditions to verify deterministic execution, correct state transitions, robust handling of mission knowledge, and safe treatment of application-level faults. Test inputs include initial status dictionaries, telemetry streams, expected commands, goals, tasks, AI-inference outputs, and reference execution traces. The objective is to establish that each core behaviour of orbital_OLIVER produces the expected internal and external effects before the software is exposed to integrated mission conditions.

The second validation layer addresses interface conformance. orbital_OLIVER is verified against the surrounding AIX execution environment to ensure that data exchange, command generation, lifecycle events, configuration updates, restart procedures, and reporting functions are correctly handled. This includes verification of interactions with the onboard software framework, payload-side processing applications, platform and payload telemetry sources, and output interfaces. The emphasis is not only on message delivery, but also on semantic correctness: commands must be issued to the correct recipient, configuration changes must be reflected in the active operational context, and execution logs must remain sufficient for ground reconciliation. Where packetised telemetry, telecommand, or data exchange mechanisms are used, the interface strategy remains consistent with CCSDS-style space data system discipline [12].

The third validation layer addresses system-level behaviour on representative hardware. This stage verifies orbital_OLIVER inside a closed-loop execution environment where autonomy, telemetry supervision, mission constraints, hosted applications, and onboard resources interact. The system is exercised through scenarios that reproduce the operational patterns expected during flight: application activation, acquisition orchestration, image-processing response handling, save/discard decisions, operational mode switching, module restart, and dynamic reconfiguration following knowledge updates. The purpose of this layer is to validate not only functional correctness, but also operational controllability: autonomous decisions must remain explainable through logs, bounded by approved constraints, and compatible with nominal execution procedures.

The fourth validation layer is the in-orbit campaign. Flight validation is organised as the final stage of the same progressive V&V chain. Each in-orbit experiment is derived from behaviours previously exercised in software and on representative hardware. The campaign targets the transition from verified functions to operational evidence: orbital_OLIVER must demonstrate that application-level autonomy can be executed

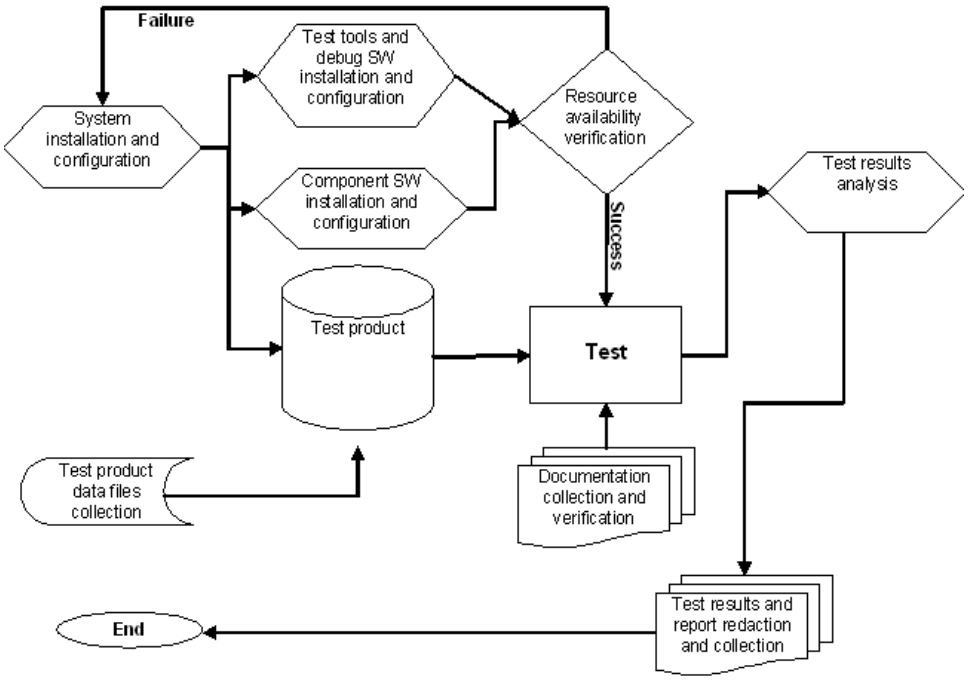

Fig. 2. Progressive V&V chain for AIX, from software-level verification to interface validation

Pillar	Scope	Relevance to OLIVER
Pillar 1	HW bring-up	execution baseline
Pillar 2	HPCP / low-level SW	Linux environment, monitoring, commanding
Pillar 3	onboard SW execution	app execution, data handling
Pillar 4	AI and autonomy	OLIVER core tests
Pillar 5	UI and E2E services	service lifecycle validation

TABLE I
MAPPING BETWEEN THE AIX VERIFICATION PILLARS AND THE ORBITAL_OLIVER VALIDATION OBJECTIVES

under real mission conditions, while preserving service continuity, safety boundaries, and ground-segment observability.

For a mission operations context, the most relevant validation properties are controllability, observability, boundedness, recoverability, and compatibility with ground supervision. Controllability ensures that orbital_OLIVER can be enabled, configured, suspended, restarted, or reconfigured through approved operational mechanisms. Observability ensures that its decisions, inputs, state changes, and outputs are available to operators through logs, telemetry, and execution reports. Boundedness ensures that autonomous actions remain inside the authorised planning envelope, including temporal, resource, payload, and safety constraints. Recoverability ensures that application-level faults can be handled through controlled restart or rollback without propagating into unsafe platform behaviour. Compatibility ensures that onboard autonomy remains aligned with the ground planning cycle and does not create an unmanaged divergence between onboard state and mission operations.

This framing is essential for AIX. The value of the campaign is not only that orbital_OLIVER can autonomously select and execute actions. The value is that these actions can be inserted into a service lifecycle that remains intelligible and governable for mission operators. This is particularly important in a Satellite-as-a-Service environment, where user requests, onboard applications, resource allocation, downlink priorities, and service-level objectives may vary over time. Autonomous behaviour must therefore be validated as part of the operational chain that connects request validation, onboard execution, result delivery, telemetry monitoring, and service closure.

The V&V approach is also aligned with established space software engineering practice. ECSS-E-ST-40C Rev.1 defines space software engineering as covering the full lifecycle of product software, including requirements definition, design, production, verification and validation, transfer, operations, and maintenance [11]. orbital_OLIVER is treated accordingly: it is not verified only as an isolated algorithm, but as product software whose behaviour must remain correct across development, integration, operational deployment, and in-flight update scenarios. This is reflected in the progressive structure of the campaign, where software tests, integration tests, hardware execution, end-to-end scenarios, commissioning activities, and flight experiments provide cumulative evidence.

The campaign is structured around a set of test objectives that map directly to the operational role of orbital_OLIVER

Test Case ID	Title	Objective	High Level Procedure(s)
1	OLIVER initialization	To verify that orbital_OLIVER can be successfully initialised.	OLIVER is turned on and it is verified that the initialisation was successful.
2	OLIVER shutdown	To verify that orbital_OLIVER can be terminated safely.	OLIVER is turned off and it is verified that all nominal operations have been completed without data loss.
3	OLIVER modules kill & recovery	To verify that if an orbital_OLIVER module fails, it can still be recovered without memory loss.	All OLIVER modules are turned off one at a time. OLIVER's watchdog brings all shutdown modules back to life.
4	OLIVER Module Manager kill & recovery	To verify that if the orbital_OLIVER module manager fails, orbital_OLIVER functionalities can still be recovered without memory loss.	The module manager is turned off. The backup watchdog brings the module manager back to life.
5	OLIVER reconfiguration	To verify orbital_OLIVER has reconfiguration capabilities whenever a new mission knowledge is uploaded.	A new configuration is loaded into OLIVER and within a pre-established time it is verified that OLIVER has reached the targret configuration while keeping its functions intact.
6	OLIVER operational mode switch	To verify that orbital_OLIVER is capable of switching operative mode when a specific command from operator is received.	A change of operative mode is commanded and it is verified that the desired configuration is achieved.
7	OLIVER payload interfaces	To verify that orbital_OLIVER is capable of receiving inputs from the payload.	For each payload data type, a test is performed to verify communication between the payload and OLIVER.
8	OLIVER platform interface	To verify that orbital_OLIVER is capable of receiving inputs from the platform.	It is tested that OLIVER is able to interface with the platform, extracting all the necessary telemetry points.
9	OLIVER output interfaces	To verify that orbital_OLIVER is capable of sending its outputs to the target subsystems.	For each output data type, it is tested that OLIVER is able to interface with the platform, correctly sending output commands.
10	OLIVER acquisition pipeline	To verify that orbital_OLIVER responds to the onboard data processing capability needs for the AIX mission.	It is tested that OLIVER correctly controls the optical payload and correctly manages the image obtained.
11	AI Inference Acceleration test	to verify access and functionality of the HW accelerator to be used for AI inference.	It is tested that inferences can be launched on the HW acceleration device and that execution speed and output data are the expected ones.

TABLE II
HIGH LEVEL TEST PROCEDURES

within AIX as described within Table II. These include initialization and shutdown, module-level restart, Module Manager restart, reconfiguration, operational mode switching, payload interface handling, platform interface handling, output interface handling, acquisition-pipeline execution, and AI-inference acceleration. Together, these tests cover the main dimensions of application-level autonomy: internal lifecycle control, interaction with mission knowledge, external interface correctness, and closed-loop orchestration of payload and processing activities.

V. CONCLUSIONS

The AIX/orbital_OLIVER campaign addresses a specific step in the maturation of onboard autonomy: the transition from autonomous functions as isolated software capabilities to autonomy as a managed component of mission operations providing verifiable evidence of an autonomy layer within a service-oriented operational architecture.

In AIX, autonomy, thus orbital_OLIVER, is evaluated in the context of a broader Satellite-as-a-Service lifecycle, where user requests, application deployment, onboard processing, payload execution, low-latency delivery, telemetry supervision, and ground planning are part of the same operational chain. This changes the validation target. The relevant question is not only whether the autonomy layer can select an admissible action, but whether that action remains bounded by approved mission knowledge, observable by operators, compatible with existing execution mechanisms, and recoverable under application-level faults.

The adopted V&V approach reflects this operational framing. Ground-based verification establishes the correctness of the autonomy loop, configuration handling, internal state transitions, and application-level fault responses. Interface validation confirms that orbital_OLIVER interacts correctly with the AIX software framework, payload-side applications, telemetry sources, and reporting mechanisms. System-level validation then exercises the same behaviours on representative hardware and closed-loop scenarios, before the in-orbit campaign assesses them under real mission conditions.

The phased IOD path across AIX-1p, AIX-1, and AIX-1+ provides the operational structure for this progression. Rather than activating autonomy as a monolithic capability, the campaign introduces and validates functions incrementally: from core software services and controlled execution, to payload integration and acquisition orchestration, and finally to richer autonomy scenarios including controlled restart, operational mode switching, and dynamic reconfiguration following mission knowledge updates.

The broader lesson for future LEO service platforms is

that onboard autonomy will not be adopted solely on the basis of algorithmic performance. It must be integrated into mission operations as a supervised, configurable, verifiable, and recoverable software function. The AIX/orbital_OLIVER campaign contributes to this transition by demonstrating how application-level autonomy can be embedded into a real service framework while preserving operational governance and ground-segment compatibility.

REFERENCES

[1] C. Lenzen, M. T. Woerle, T. Göttfert, F. Mrowka, and M. Wickler, "Onboard planning and scheduling autonomy within the scope of the firebird mission," in *SpaceOps 2014 Conference*, 2014, p. 1759.

[2] S. Chien, I. Zilberstein, A. Candela, D. Rijlaarsdam, A. Perrocheau, A. Dunne, T. Hendrix, O. C. Grauc, A. G. i Mestrec, M. P. Bovec *et al.*, "Flight of dynamic targeting on cognisat-6–update," in *International Conference on Space Operations*, 2025.

[3] B. Ruszczak, K. Kotowski, D. Evans, and J. Nalepa, "The ops-sat benchmark for detecting anomalies in satellite telemetry," *Scientific Data*, vol. 12, no. 1, p. 710, 2025.

[4] V. Fortunato, L. Amoruso, S. Antonetti, C. Abbattista, G. Furano, L. Feruglio, M. B. Marco, and S. Oliva, "Reshaping the earth-observation value chain through ai-express powered satellite-as-a-service," in *2025 European Data Handling & Data Processing Conference (EDHPC)*. IEEE, 2025, pp. 1–4.

[5] ESA, "AI-express InCubed," https://incubed.esa.int/portfolio/aix/, [Online; accessed 30-April-2026].

[6] Planetek Italia, D-Orbit, AIKO, "AI-eXpress Project Page," https://www.aiexpress.eu/project, [Online; accessed 30-April-2026].

[7] Planetek Italia, "AIX Smart In-Orbit Data Processing," https://www.planetek.it/en/projects/aix_smart_in_orbit_data_processing, [Online; accessed 30-April-2026].

[8] AIKO, "orbital_OLIVER Autonomous mission operations, directly onboard," https://www.aikospace.com/solutions/oliver, [Online; accessed 30-April-2026].

[9] Planetek, "Third AI-eXpress satellite in orbit for a new era of edge intelligence in space," https://www.planetek.it/en/news_events/news_archive/2025/11/third_ai_express_satellite_in_orbit_for_a_new_era_of_edge_intelligence_in_space, 2025, [Online; accessed 30-April-2026].

[10] A. Hein and C. BRUCE ROSETE, "Space-as-a-service: A framework and taxonomy of-as-a-service concepts for space," in *International Astronautical Congress 2022*, 2022.

[11] European Cooperation for Space Standardization, "European Cooperation for Space Standardization, ECSS-E-ST-40C Rev.1 Software, Apr. 30, 2025," https://ecss.nl/standard/ecss-e-st-40c-rev-1-software-30-april-2025/.

[12] Consultative Committee for Space Data Systems, "Space Packet Protocol, CCSDS 133.0-B-2, Issue 2, Jun. 2020," https://ccsds.org/publications/allpubs/entry/3264/.