# Onboard Planning and Scheduling for Autonomous Satellite Operations Towards Multi-Agent Cooperation

Riccardo Maderna
AIKO Srl
Turin, Italy
riccardo.maderna@aikospace.com

Marco Morgese
AIKO Srl
Turin, Italy
marco.morgese@aikospace.com

## ABSTRACT

This paper presents an onboard planning and scheduling pipeline designed to enable autonomous satellite operations. We first provide a system-level overview of the architecture, highlighting how the planning agent ingests state and contextual information and issues time- and resource-constrained commands in real time. We then discuss how interaction with other onboard applications - payload data processing, health monitoring, orbit maintenance, and cooperative tasking - can enable the highest degree of operational autonomy. The behavior of the autonomous agent is exemplified through a set of simulation scenarios that illustrate its ability to adapt to varying mission conditions, evolving operational constraints, and off-nominal events. Finally, we introduce a concept for managing multi-agent coordination, focusing on task assignment. This work shows a concrete implementation of onboard satellite autonomy which enables more resilient, scalable, and efficient autonomous operations in current and future space missions.

## KEYWORDS

Autonomous spacecraft, Mission operations, Planning and scheduling, Multi-agent cooperation, Task assignment

**ACM Reference Format:**
Riccardo Maderna and Marco Morgese. 2026. Onboard Planning and Scheduling for Autonomous Satellite Operations Towards Multi-Agent Cooperation. In *Appears at the International Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., Paphos, Cyprus, May 2026*, IFAAMAS, 6 pages.

## 1 INTRODUCTION

The growing commercial competition in space is driving the need for more efficient and scalable missions. Advanced autonomy is emerging as a key enabler, supporting the management of increasingly complex space systems, improving mission output, cutting operational costs, and enabling new mission concepts [14]. Enhanced onboard processing now allows intelligence to be embedded directly in spacecraft, and the tight integration of onboard and ground operations improves responsiveness, especially for user-driven missions such as Earth observation. This integration enables just in time tasking and continuous schedule optimization, making the system more adaptable and responsive to dynamic requirements. Onboard autonomy also unlocks event driven mission concepts that depend on unpredictable phenomena.

Traditional onboard planning has focused mainly on reactive repair of ground generated schedules, as seen in systems like VAMOS [15] , but purely reactive approaches [4, 7] can be short-sighted and suboptimal. Conversely, fully proactive planning offers long term reasoning but lacks flexibility when unexpected events occur, usually requiring full re-plan [1]. The most effective strategy is therefore a hybrid approach that combines long-term planning with reactive adjustments. This work describes an integrated onboard planning and scheduling pipeline enabling full goal-oriented autonomy. A proactive component generates a long horizon flexible plan, while a reactive component executes it, absorbs uncertainties, and refines actions using real time information. A reasoning engine monitors plan validity and triggers re-planning when needed. This system combines the timeline formalism [3] and conditional procedures, providing an effective framework to manage concurrent activities, constraints, resources and events, and it is designed for broad applicability across missions.

As autonomy scales to constellations, the ground segment primarily takes on the role of allocating user requests among satellites with different capabilities, while each spacecraft autonomously plans and executes its assigned tasks. This aspect is explored in Section 5 of the paper.

## 2 ONBOARD PLANNING AND SCHEDULING SYSTEM

The architecture of the planning and scheduling agent (Figure 1) is composed of three primary components: Request Manager, Planning Manager, and Execution Manager. The Request Manager is responsible for managing incoming user requests. It monitors the life cycle of each request from submission to completion and assigns priority levels based on mission rules and operational constraints to ensure that the system processes requests in an optimal order.

The Planning Manager provides the proactive component: it generates long-term flexible plans of activity using a timeline-based approach. It incorporates user requests, mission goals, operational constraints, and mandatory activities received from operators. The resulting flexible plan can be seen as an envelope of deterministic schedules, each one representing a fixed-in-time sequence of actions or commands. This flexibility allows for absorbing uncertainties during execution without the need for re-planning, which strongly reduces the computational load of the whole planning pipeline and increases the predictability of the satellite's behavior.


*Appears at the International Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., S. Chien, G. Picard, I. Zilberstein (Chairs), May 2026, Paphos, Cyprus.* © 2026 Copyright held by the owner/author(s).

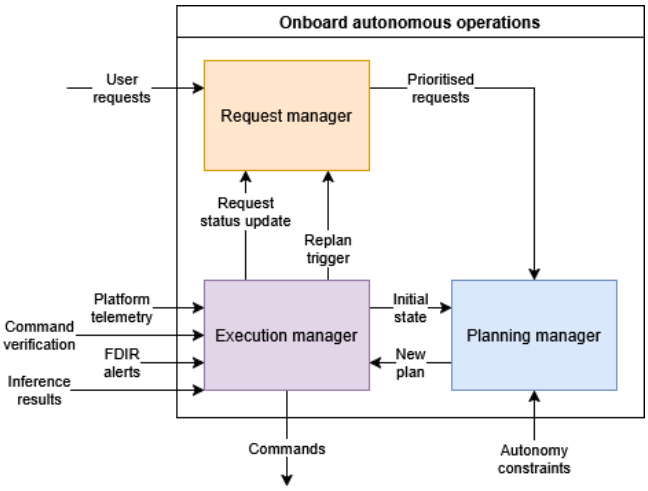



**Figure 1: High-level system architecture.**

The Execution Manager translates planned activities into a sequence of commands via conditional procedures. Flexibility of the plan and conditional branches are exploited to optimize the mission return and to adjust the schedule in response to just-in-time information (like edge data processing results) and contingencies. The Execution Manager also monitors the validity of the plan and triggers a re-planning process in case of: violation of resource availability envelopes, indicating a mismatch between forecast and actual resource use; occurrence of anomalies that invalidate the plan; reception of new information, including new user requests or mandatory activities from ground operators.

Both Planning and Execution Managers implement generic engines that are highly configurable to mission requirements ensuring high re-usability.

## 2.1 Proactive planning

The Planning Manager is based on timeline-based planning [3], a formalism well suited for mission planning thanks to its expressiveness and natural alignment with how space mission constraints and resources are modeled. In this approach, the world is represented through state variables whose values evolve over time. These variables may be external (e.g., ground station visibility) or internal (e.g., spacecraft orientation). Their evolution is encoded in timelines, which are sequences of tokens representing flexible time intervals during which each variable holds a specific value. A chronicle aggregates all timelines along with constraints and resource information.

The planner searches in the space of chronicles, starting from a partial one and iteratively resolving flaws (i.e., constraint violations) until a complete, feasible solution is found. The resulting plan consists of timelines and their mutual constraints. Since constraints are only inserted to ensure feasibility, the obtained plan maintains margins for flexibility that is exploited at execution time. To handle both controllable and uncontrollable temporal relations, each chronicle maintains a Simple Temporal Network with Uncertainty (STNU), which models uncertain durations, precedence constraints, and synchronizations with external events. The STNU must remain consistent and dynamically controllable to ensure executable plans [6]. Resource feasibility is also maintained through dedicated managers for binary, capacity, and consumable resources. A pseudo-code overview of the planning routine is the following:

**Algorithm 1** Plan(initial_state, final_state, goal_list

```
1:  function PLAN(initial_state, final_state, goals)
2:      chronicle ← INITIALIZE(initial_state, final_state)
3:      exploration_stack ← ∅
4:      while chronicle is flawed do
5:          flaw ← GETHIGHESTPRIORITYFLAW(chronicle, goals)
6:          resolvers ← COMPUTERESOLVERS(flaw)
7:          if |resolvers| > 1 then
8:              ADDDECISIONPOINT(exploration_stack)
9:              resolver ← GETBESTRESOLVER(resolvers)
10:             ADDRESOLVER(exploration_stack, resolver)
11:         end if
12:         APPLY(chronicle, resolver)
13:         if chronicle is inconsistent then
14:             backtrack to the previous decision point
15:         end if
16:     end while
17:     return EXTRACTPLAN(chronicle)
18: end function
```

The planner supports two goal types: permanent and opportunity goals. Permanent goals cover things like keeping batteries charged or maximizing data acquisition and can adapt to changes in mission priorities. Opportunity goals refer to time bound tasks, such as targeted imaging requests, which are provided as a prioritized list by the Request Manager, and the planner selects the best subset that can coexist with other activities while preserving feasibility.

## 2.2 Reactive execution loop

The Execution Manager implements reactivity at two levels. First, it executes the flexible plan by assigning fixed times at time-points in the STNU. Firings of controllable time-points are chosen to optimize mission return; uncontrollable ones are based on sensed events or the conclusion of procedures.

The second level is the implementation of activities via conditional procedures. It is supported by a procedure execution engine that interprets procedures written in a compact domain-specific language. The engine processes statements sequentially - evaluating expressions, assignments, control flow, and command statements - until it encounters a wait condition that is not satisfied. When state changes and the wait condition is satisfied, execution resumes. Conditional branches and wait conditions can be evaluated based on internal procedure variables and mission state parameters, like satellite telemetry or the state of dispatched commands.

# 3 INTERACTIONS WITH ONBOARD APPLICATIONS

Onboard planning beats ground mission planning as it can exploit real-time information. Therefore, it becomes more powerful in combination with other onboard applications that can provide the same inputs ground operations teams use for mission planning, but

with low latency. By doing so, it is possible not only to increase mission efficiency but to unlock new, highly dynamic, mission concepts. Specifically, four classes of supporting applications are considered in this paper:

- **Onboard payload data processing:** applications able to process payload data to produce actionable information on board, like event detections. Such insights are exploited by the autonomous operations agent in two ways: they are input to evaluate conditional branches of procedure (for instance, discarding low quality data and abort further processing) and they are used to generate requests onboard (for instance, follow-up observations following fire detection).
- **Spacecraft health monitoring:** applications able to detect and classify anomalies, propose recovery or mitigating actions, and monitor the effect of the recovery plan execution [9, 13]. It produces two types of data: i) alerts that inform the autonomous agent to avoid usage of unavailable resources or operating modes; ii) recovery plans that are inserted in the plan as autonomy constraints.
- **Autonomous orbit maintenance:** applications able to compute station keeping and/or collision avoidance maneuvers [2, 5, 12], which are autonomy constraints for the autonomous operations agent.
- **Cooperative tasking:** application that enables distributed and cooperative tasking among satellites in a constellation endowed with inter-satellite links. It provides a source of new user requests and a way to re-distribute requests that the satellite is unable to satisfy efficiently.

## 4 EXAMPLES OF AGENT BEHAVIOR

An example scenario illustrates the algorithm's behavior and its advantages over common satellite planning methods. The case concerns a small Earth observation mission whose primary goal is acquiring user requested targets, while secondary wide area acquisitions increase mission return. Onboard data processing (e.g., cloud detection) helps adjust acquisition frequency to save resources.

The planning problem includes two external timelines - eclipses and ground station visibility - and two internal timelines – operating mode (feasible states: idle, observation, monitoring, downlink, ground control) and spacecraft orientation (feasible states: sun pointing, antenna to ground, camera to ground, slewing). Battery level and memory usage are treated as consumable resources. User requests are modeled as opportunity goals; extra data takes as a permanent goal. At execution level, conditional procedures describe that the observation activity can switch between high and low frequency acquisition depending on local conditions, while monitoring can escalate to observation when beneficial.

Figure 2 shows an example of the planning process. Initially (Fig. 2a), the external timelines are fully defined (in blue), while internal ones contain only initial states and mandatory maintenance windows. Red tokens indicate unsupported elements. Flexible time intervals appear in lighter colors. A worst-case battery profile is shown at the bottom. Fig. 2b shows an intermediate state after evaluating opportunity goals: some are excluded due to conflicts, resource limits, or inconsistencies. Fig. 2c presents the output plan, with all tokens supported and remaining flexibility, which depends on scenario uncertainty (e.g., activities with uncontrollable durations).

Figure 3 highlights reactive execution behavior. The first row shows an opportunity goal, leading to a planned observation over an area with some flexibility before switching to monitoring. Expected and actual battery profiles are shown with dashed and solid lines. The bottom row shows the reactive loop's refinements. In Fig. 3a, ongoing observation is executed at high frequency. The mode adjusts dynamically based on cloud coverage until the opportunity goal ends (Fig. 3b). Low frequency acquisitions consume less battery, generating a surplus; the execution loop therefore extends observation to pursue permanent goals (Fig. 3c). When battery or memory exceeds thresholds, a new local goal triggers a switch to the next activity (Fig. 3d).

## 5 MULTI-AGENT COOPERATION

Constellations are a use case of particular interest for autonomous satellites as they allow ground segments to operate a large number of spacecraft at a fraction of the effort. No detailed planning is performed on the ground, which must only distribute user and onboard generated service requests. This section describes a multi-agent negotiation system concept for the task distribution problem.

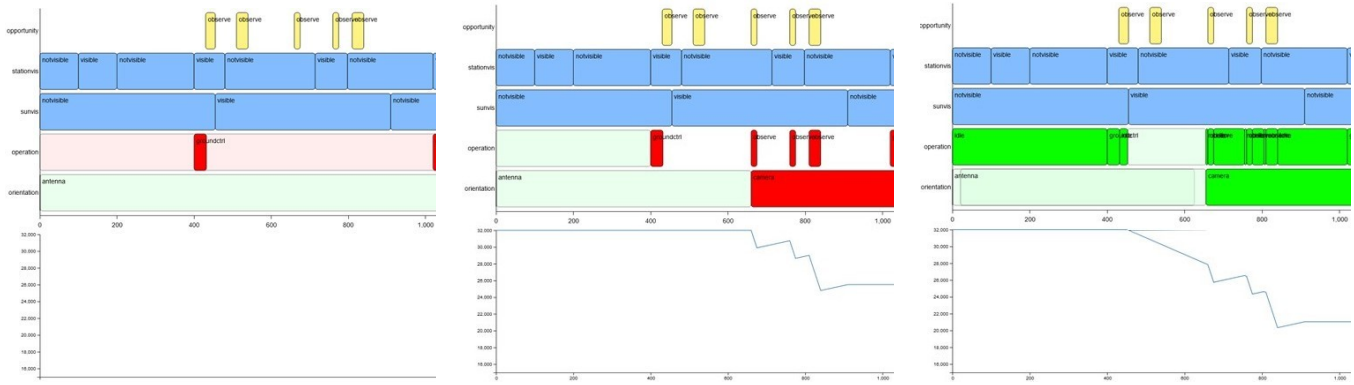
The ground system collects user and onboard generated service requests. When a satellite comes into contact with a ground station, a subset of the pending requests is assigned to it, depending on the satellite's capabilities and availability. The problem can be modeled as an auction with the ground segment as the auctioneer and satellites in the constellation as negotiating agents. The challenge is that satellites contact the ground segment at different times, making it impossible for all agents to place their bids. Therefore, the ground segment maintains a proxy representation of agents, which participate in auctions. Every time a satellite communicates with the ground, the state of the corresponding proxy is updated, and an auction for pending tasks is run among all proxies. Figure 4 gives a pictorial view of the system architecture, whereas the sequence diagram in Figure 5 describes the auction process triggered by a new satellite contact.

Each proxy is a tuple $p_i = (R_i, L_i, K_i, \Pi_i, \mathcal{K}_i, \Delta_i, \gamma_i)$ where:

- $R_i$ is the set of assigned pending requests,
- $L_i$ is the set of resource level arrays indicating the predicted resources levels for the various satellite resources at future time instants,
- $K_i$ is the set of services that satellite can currently provide,
- $\Pi_i$ is the set of payloads available onboard the satellite to provide services,
- $\mathcal{K}_i : K_i \rightarrow \mathcal{P}(\Pi_i)$ is a function that maps each service $k \in K_i$ to the subset of payloads required to perform the service,
- $\Delta_i : K_i \rightarrow \mathbb{R}$ is a function that maps each service $k \in K_i$ to the amount of time required to perform the service,
- $\Gamma_i : K_i \rightarrow \mathbb{R}^2$ is a function that maps each service $k \in K_i$ to the amount of resources used to perform the service.

The main steps of the process are:

(1) **Proxies update**. The state of the Proxy of the communicating satellite is updated with real data, while the other Proxies are updated through simulation according to their expected evolution.



(a) Initial Chronicle. (b) Intermediate status. (c) Final flexible plan.

**Figure 2: Example of proactive planning process from initial Chronicle (portion) to final plan. The top row (yellow) shows pending requests to plan for; blue marks external timelines; red and green mark unsupported and supported internal tokens, respectively. Flexible time intervals appear in lighter colors. The bottom graph describes the worst-case battery profile.**

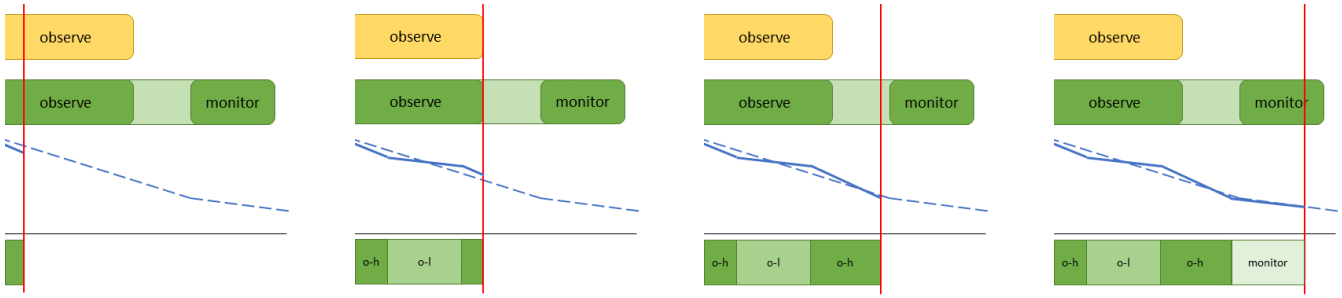



**Figure 3: Example of reactive execution process. An observation request (yellow) has been inserted in the plan (green). (a) Ongoing observation is executed with high frequency acquisitions; (b) the mode adjusts dynamically based on cloud coverage; (c) plan flexibility allow to continue high-frequency observations to exploit the battery surplus; (d) when battery lowers, a new local goal triggers the next activity.**

(2) **Auction**. The Auctioneer broadcasts a set $R$ of pending requests (default strategy for selection is FIFO, but it can be tailored on mission requirements) and Proxies answer with a bid for each request. The Auctioneer ranks the bids and assigns the highest-valued requests to the winning Proxies, which updates their state. The number of announced and assigned requests is configurable, and this step can be performed multiple times to assign all requests.

(3) **Request transmission**. Once the assignment process has been completed, a message is sent to the communicating satellite before the end of the visibility window, containing the new requests assigned to it since the last contact.

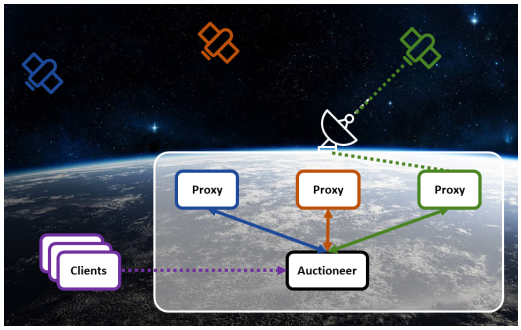



**Figure 4: Pictorial view of negotiation system.**

## 5.1 Bid generation

Upon reception of an announcement, each Proxy generates a bid for every announced request, which expresses the capability of the satellite to satisfy the request timely and efficiently. Bids are computed independently for each request in the announcement.

The Proxy $p_j$ does not bid if the satellite is unable to provide the requested service or it has no visibility on the target before the request expiration date; otherwise, it bids a convex combination of $N_b$ bidding terms $\gamma_n \in [0, 1]$: $b_j = \sum_{n=1}^{N_b} w_n \gamma_n$

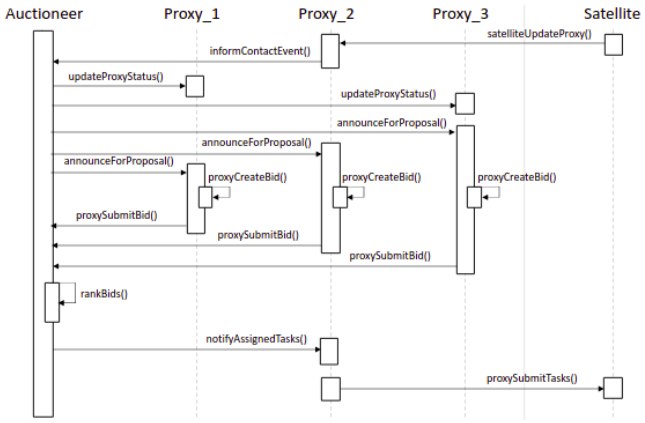



**Figure 5: Auction process triggered by a new satellite contact.**

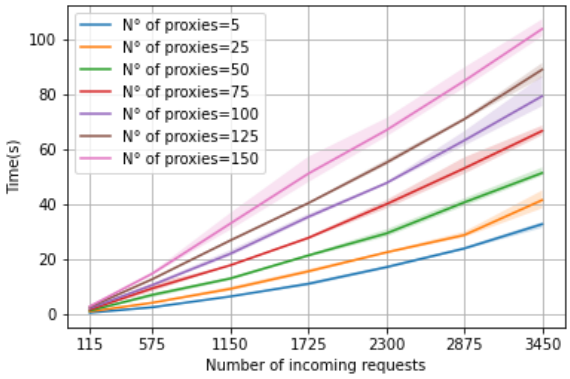



**Figure 6: Auction time for single-item auctions.**

In this first proof of concept the considered bidding terms are:

- **Assigned requests:** the bid decreases with the number of already assigned requests.
- **Request priority:** the bid increases with the priority and the approaching expiring date $t^{exp}$ of the request. This term is independent of the Proxy status but serves as an incentive to assign high priority requests first.
- **Request satisfaction time:** the bid is higher the shorter the time required for the satellite to execute the requested service and transmit the data back to ground.
- **Satellite availability:** when conflicting requests are assigned to the same satellite, it results in a delay in the satisfaction of all but the highest priority request. Therefore, the higher the number and priority of conflicting requests the lower the bid.
- **Satellite resources:** the bid is lower the less resource availability of the satellite.

## 5.2 Preliminary analysis

The reference scenario for preliminary analysis was an Earth Observation constellation in LEO with no inter-satellite links. Satellites in the constellation are equipped with optical, SAR or both payloads, with a total of four different services that can be provided (optical image, SAR spot, SAR strip, optical+SAR combined acquisition). Simulations have been run with random initialization on an off-the-shelf laptop with Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz and 12 GB of RAM.

Time complexity analysis was performed for increasing constellation size (up to 150 satellites, with uniform distribution of satellite type) and user requests per episode (up to 3450, with distribution of service type inversely proportional to resource requirements). Figure 6 shows the results for single-item auctions demonstrating the feasibility of running auctions well within the duration of a ground contact window. Experiments have been also conducted by changing the announcement size and assignment size: larger announcement size allows agents to bid for the best requests first for increased computational time, larger assignment size reduces the number of auction rounds.

Sensitivity analysis has been conducted to evaluate the behavior against changing weights of the bid terms. Considered cases were: uniform relative weight, ablation cases with one null weight, opposite cases with one weight set to a high value. Analysis explored impact on load balance (variance in number of tasks assigned to satellites), the number of conflicting tasks assigned to each satellite, and bid evolution over the auction rounds (both in terms of overall value and for each bid term).

## 6 CONCLUSION

The present work described a proactive-reactive approach for onboard automated planning. The solution is highly flexible and can be applied to a wide variety of missions. The proactive loop allows for long-term, goal-oriented planning while the reactive loop enables local decision-making and plan refinement in view of uncertainty and contingencies. Parts of the pipeline have been tested in testbed (Tyvak International, D-Orbit) and in-orbit demonstrations between 2023-2026 (D-Orbit [10], AI-Express project [8, 11]), whereas a demonstration of the complete agent is expected in 2027 (In-Orbit Space Lab project, others TBC).

Extension to the multi-agent scenario requires further study to address two main aspects. First, the impact of deferred proxy updates on task assignment performance, which becomes even more critical when satellite-to-satellite communication is enabled. Second, the interaction between the planning agent and the onboard cooperative tasking application, which have been for now treated separately.

## REFERENCES

[1] M. Troesch et al. 2020. MEXEC: An Onboard Integrated Planning and Execution Approach for Spacecraft Commanding. In *International Conference on Automated Planning and Scheduling (ICAPS 2020)*. 9.
[2] S. Nag et al. 2021. Prototyping operational autonomy for space traffic management. *Acta Astronautica* 180 (2021), 489–506.
[3] M. Ghallab, D. Nau, and P. Traverso. 2016. *Deliberation with Temporal Models*. Cambridge University Press. 114—-154 pages.
[4] E. Herz, D. George, T. Esposito, and K. Center. 2014. Onboard Autonomous Planning System. In *SpaceOps 2014 Conference*.
[5] J. Kruger, S. S. Hwang, and S. D'Amico. 2024. Starling Formation-Flying Optical Experiment: Initial Operations and Flight Results. *preprint arXiv:2406.06748* (2024).
[6] P. Morris, N. Muscettola, and T. Vidal. 2001. Dynamic control of plans with temporal uncertainty.

[7] C. Powell and A. Riccardi. 2022. On-board re-planning of an earth observation satellite for maximisation of observation campaign goals. In *73rd International Astronautical Congress (IAC 2022)*. 11.

[8] AIKO Srl. [n.d.]. AIX - Smart In-Orbit data processing. https://www.aikospace.com/blog/aix-smart-in-orbit-data-processing [Accessed: 2026-04-10].

[9] AIKO Srl. [n.d.]. gifted_GENE. https://aikospace.com/products/gifted-gene [Accessed: 2026-04-10].

[10] AIKO Srl, D-Orbit Spa, and Unibap. [n.d.]. Successful completion of the In-Orbit Demonstration campaign for orbital_OLIVER.

[11] Planetek Italia Srl. [n.d.]. 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 [Accessed: 2026-04-10].

[12] F. Toussaint, J. Thomassin, and S. Laurens. 2022. ASTERIA in-orbit testing on OPSSAT: an on-board autonomous orbit control solution including collision risks avoidance.

[13] A. Wander and R. Förstner. 2013. Innovative Fault Detection, Isolation and Recovery Strategies On-Board Spacecraft: State of the Art and Research Challenges. *Deutsche Gesellschaft für Luft- und Raumfahrt* (2013).

[14] D. Wang, J. A. Russino, C. Basich, and S. Chien. 2022. Analyzing the Efficacy of Flexible Execution, Replanning, and Plan Optimization for a Planetary Lander. In *Proceedings of the Thirty-Second International Conference on Automated Planning and Scheduling*, Vol. 32. 9.

[15] M. T. Wörle and C. Lenzen. 2013. Ground Assisted Onboard Planning Autonomy with VAMOS.