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

SpaceOps-2025, ID #500

A flexible and extensible Mission Planning System applied to Space-Based Space Surveillance and Tracking

Guillermo Janner Acero[a]*, Florian Strasser[a], Elias Lerch[a], Pietro Silvagni[a], Stefan Frey[a]

[a] ***Vyoma GmbH, Germany***

*** Corresponding Author, guillermo.janner@vyoma.space**

Abstract

In recent years, the number of satellites organized into medium- and large-sized constellations has grown significantly. Due to rapidly evolving hardware, these constellations often comprise heterogeneous platforms, creating challenges for mission-specific and routine operations. To address these challenges, the need arises for a flexible mission planning system that is extensible, configurable, and capable of scheduling tasks across diverse hardware platforms. Additionally, as constellations grow in size, autonomous operations become increasingly essential.
To meet these requirements, Vyoma has developed an in-house Mission Planning System designed to orchestrate a constellation of space-based observers. This system enables the establishment of Europe's first sovereign space-based surveillance network, capable of mapping all space objects in Low-Earth Orbit.
This paper demonstrates how this flexible mission planning system can accommodate various constellation sizes and mission types using a novel activity pattern. The system is also designed to be scalable, with the capability to be deployed locally on-premise or as a Software-as-a-Service (SaaS) platform. This approach not only broadens its usability for other users but also reduces development and maintenance costs.

Keywords: Satellite Operations, Ground Segment, Mission Planning, Timeline, Scheduling

1. Introduction

With an ever-growing number of objects in Earth orbit, there is a clear need to safeguard the existing assets in space as well as ensure the long-term usability of this environment. This requires an up-to-date catalogue of all of these objects around Earth, together with a suite of tools capable of propagating their trajectories against each other in order to detect and react on potential conjunctions in a timely manner. To build this catalogue, Munich-based space situational awareness provider Vyoma is developing its own constellation of satellites, the *Flamingo* constellation, representing a significant endeavor aimed at delivering an affordable and highly available depiction of Low-Earth Orbit (LEO) activities. The objective is achieved through in-situ surveillance—utilizing telescopic sensors mounted on satellites that dynamically align to capture passing objects of interest. This approach requires optimal asset utilization, characterized by achieving high duty cycles for the cameras to ensure continuous operational status and data generation. Hence, Vyoma developed an in-house solution which will soon be demonstrated during operations of *Flamingo-1*, the first of twelve satellites in their constellation.

To maximize sensor efficiency many factors have to be taken into account, such as target priority, predicted signal-to-noise ratios, opportunity costs, and constraints imposed by satellite capabilities related to power, memory, and communication. This necessitates an integrated Mission Planning System (MPS)

capable of sophisticated optimization strategies for day-to-day observation target selection and data acquisition.

The envisioned expansion of the *Flamingo* constellation to 12 satellites aims to enhance observational frequency, targeting observations every 45-50 minutes for most objects in LEO, with minimum latency for critical observations. The ground segment mission architecture must be scalable and seamlessly integrate diverse satellite platforms while minimizing human operational involvement.

This paper is structured as follows: Section 2 outlines the most important requirements. Section 3 describes the overall system architecture and core design concepts, introducing the domain model, its interfaces and adapters, use cases and illustrates how Activities are the core of this novel design. The following Section 4 introduces the chosen concepts with respect to the automation of satellite operations. Section 5 describes the steps taken to integrate the space-based space surveillance *Flamingo* mission into the presented operations environment. Finally, Section 6 draws a conclusion to the work presented within the scope of this paper and Section 7 provides an outlook to future developments for Vyoma's MPS.

2. Requirements

In order to realize the mission planning aspect of this vision, the following set of requirements has been derived in order to provide a comprehensive overview of the key aspects:

- **Mission Flexibility and Scalability:** The MPS shall offer an autonomous spacecraft operations service that is both flexible, with respect to mission characteristics and its specific goals, and scalable, with respect to the number of constellations and assets per constellation.
- **Constellation Heterogeneity and Management:** The system shall be capable of managing satellite constellations that differ in their design e.g. different spacecraft bus or payload designs as well as size. These constellations are required to be re-configurable at all times, including but not limited to adding or removing assets, as well as defining constellation-wide mission targets as well as individual targets on spacecraft level.
- **Optimized Mission Planning:** The mission planning component of the system shall encompass multiple approaches which collectively aim to achieve measurably improved mission operations. The optimization efforts of the MPS include but are not limited to:
  - *Observation Planning:* Find, evaluate and select observation opportunities utilizing various payload operational modes based on the respective mission specific constraints and targets.
  - *Contact Planning:* Compile a schedule of contacts, whether to stations on ground or other means of relaying data to and from the spacecraft, based on configurable metrics concerning the amount of data, and latency requirements related to maintaining stable platform operations and at the same time offering timely delivery of collected payload products.
  - *Manoeuvre Planning:* Employ optimal manoeuvre calculations in order to not compromise mission lifetime through propulsion overconsumption. On top of that, the optimized manoeuvre calculations, shall, in combination with the other optimization aspects, allow decisions closer to the projected events, and therefore benefit from a lower level of uncertainty.
  - *Resource Utilization:* Optimize the allocation of resources on-board the spacecraft towards a specific combination of available spacecraft operations, that achieves the largest contribution towards the goals of the mission.
- **Autonomous Operations**: Achieving full automation to reduce operational costs while allowing for manual intervention during contingencies or urgent requests. The goal is to eliminate the need for compiling operations schedules allowing to focus on advancing the mission concepts.

- **Human Operator Interface:** The system shall be reactive to request-based inputs from spacecraft operators, comparable to an incremental planning approach as presented by Wörle in [1]. These interactions are designed to decrease the manual effort and time spent in recurring procedures that offer a high potential for automation and therefore free significant quantities of operator capacities.
- **Standard Compliance:** The system shall comply with Consultative Committee for Space Data Systems (CCSDS) standards to enhance data exchange to other system and external partners. Therefore it shall only use the formats defined in [2], such as Attitude Ephemeris Message (AEM), Conjunction Data Message (CDM), Orbit Ephemeris Message (OEM).
- **System Integration and Extensibility**: The system shall be extensible by other already existing systems provided by manufacturers or already developed, such as a Flight Dynamics System (FDS), Command and Control (C2) or Ground Station Network (GSN).
- **Software as a Service (SaaS) and Security:** The system shall support seamless deployment and scalability across both cloud-based and on-premise environments, while conforming to modern cyber security best practices. These include end-to-end data encryption (at rest and in transit), secure authentication and authorization, audit logging, vulnerability and patch management, and compliance with internationally recognized standards such as ISO/IEC 27001, NIST SP 800-53, as well as domain-specific guidelines from the Consultative Committee for Space Data Systems (CCSDS) and the European Cooperation for Space Standardization (ECSS), where applicable.

3. System Architecture

To fulfil the MPS requirements an architecture and data model have to be chosen which are extensible and highly configurable. Multiple architectures have been proposed and implemented for this in the past. Most of which are purpose built [3], [4] or targeting only a subset of the presented requirements. We chose to use a similar approach as proposed by [1], but adapting it rigorously to principles of clean [5] and hexagonal [6] architecture. This architecture, also known as Ports and Adapters, promotes a clear separation between the core business logic and external systems by defining explicit interfaces for interaction. This decoupling enhances modularity, facilitates testing by enabling the use of mocks and stubs, and supports maintainability by allowing infrastructure components to evolve independently. Furthermore, it fosters reusability of the domain logic across multiple delivery mechanisms, leading to more robust and adaptable software systems. This adaptability is key for integration of systems such as the GSN where we might have different kind of networks for different satellites.

The essential building blocks of this architecture are the domain model, domain services, domain interfaces, use cases and adapters, see Figure 1. The domain model, services and interfaces are the reusable components allowed to be used across all other components. They were identified using Domain-driven design principles [7]. Adapters are concrete implementations of domain interfaces composed by domain services and models and can depend on other domain interfaces. Adapters are not allowed to use other adapters directly, but only through interfaces. The rules for this architecture is enforced during compile time. An adapter covers exactly one technical (e.g. database access, exposing APIs) or functional aspect (e.g. mission specific). For reactive use case invocation and event distribution an event bus is integrated. Refer to Figure 2 for an overview of the instantiated architecture for the Flamingo mission.

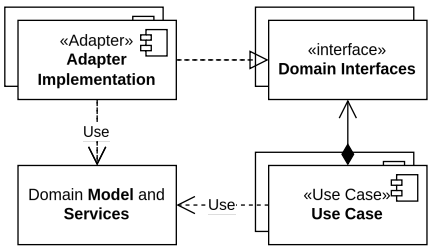

Figure 1: UML class diagram of the essential building blocks and allowed dependencies and interactions of the hexagonal architecture.

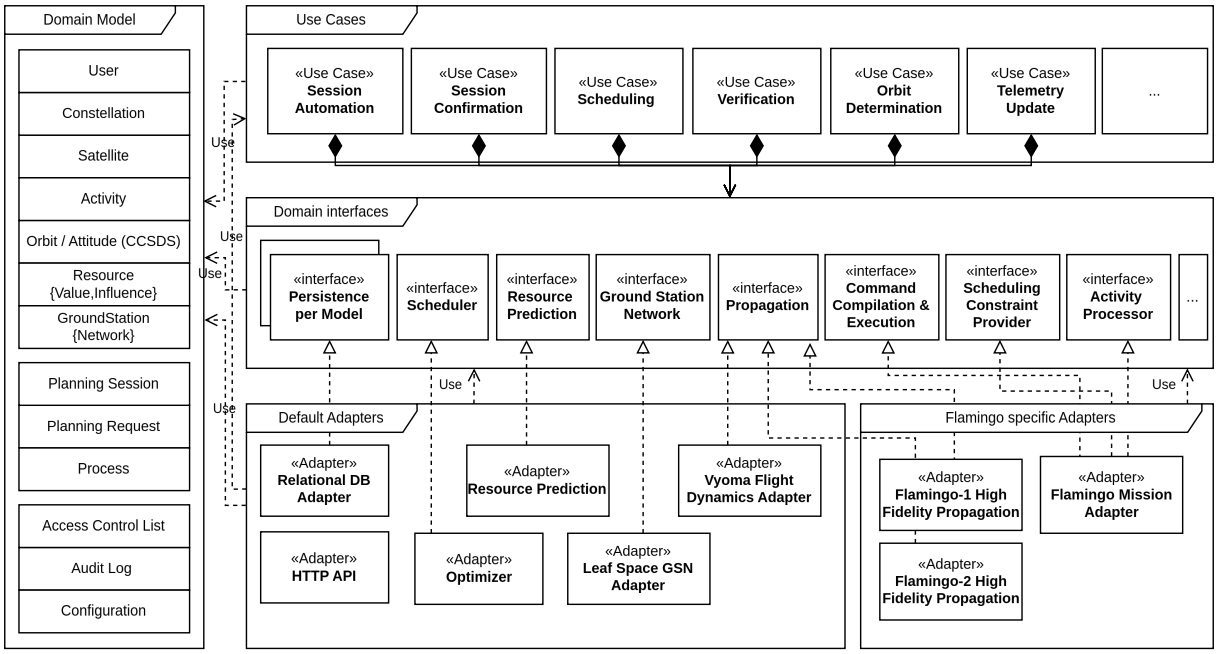

Figure 2: Instantiated hexagonal architecture overview including Vyoma Flamingo specific implementation.

3.1. Domain Model

The following core entities for the domain model were identified. For a comprehensive list, refer to Figure 2. Note that the domain model has to be always agnostic to mission specifics.

- **Satellite**: Representation of a physical asset in space. A satellite is attached to an organization. Multiple satellites can be managed as a group by a Constellation model.
- **Activity**: Denotes a capability of a satellite or constellation (e.g., manoeuvre, ground station contact). Activities are specific to a satellite platform or mission and allow custom implementations using the strategy pattern [8]. Each activity is executed by the stages of an activity processor, which defines the primary planning workflow. Refer to Section 3.4 for details about how activities can be implemented and are processed as this is the core workflow of the system.
- **Process**: An instantiation of an activity, characterized by a specific start time and duration (e.g., perform maneuver at timestamp t_0). Processes are the immutable outputs of the scheduling use case.
- **Planning Request**: Represents the mechanism to express mission and operational intents to the system. Each request is activity-specific and results in the generation of *0..n* processes. Planning requests have a *state* determining whether the request is e.g. `PENDING` for execution, currently being `EXECUTED`, `FAILED` or `FULFILLED` to allow user and operator feedback. Also, planning requests allow generic arguments which have to match a definition given by its Activity. E.g. the arguments for a manoeuvring request would be the manoeuvre itself like time, thrust direction, magnitude etc.
- **Planning Session**: Defines the planning horizon, which consists of a collection of processes. A scheduling use case execution optimizes the session. A session has a *state*, e.g. `CREATED`, `OPERATIONAL`, `PENDING_APPROVAL`. Many sessions can coexist, e.g. for evaluation of different scenarios. However, only one `OPERATIONAL` session can exist at a time.
- **Configuration**: Generic key value entities that contain configuration values to be used across the entire system. They can be specific however per use case, adapter or even specific only for a satellite or constellation allowing a high degree of flexibility in configuration.
- **Resource**: Representation of a physical or virtual resource and its associated values as a time series. Can be used to represent e.g. power generation, manoeuvring budget, Global Navigation Satellite System (GNSS) measurements, subsystem states or parameters like drag coefficient and the radiation pressure coefficient. It is possible to store values of type `MEASURED` and `ESTIMATED` (e.g. as a result of an orbit determination).

- **Resource Influence**: Describes the specific effect of an activity on a resource. For each instantiated process, resource influences can be estimated. The current implementation supports piecewise-defined polynomials.
- **Orbital information**: Orbital information are stored as CCSDS compliant files [2]. Three kinds of orbital data exist: Type `DETERMINED` with no session assigned which are results of orbit determination. The latest orbital data in this state are used as base for every new planning session. Type `ESTIMATED` with a planning session assigned, being the result of a planning session optimization. Finally, type `ESTIMATED` and no session assigned. The latest orbital data of this kind is considered the operational orbit as it includes the latest known planned processes (such as manoeuvres).

3.2. Domain Interfaces and Adapters

To enable a high degree of flexibility, small and self-contained domain interfaces have been identified. These include: Per Model Write and Read access, Session Optimization (Scheduling), Optimization Constraint Provider, Resource Prediction, Ground Station Network Provider, Orbit Propagation, Orbit Determination, Event Detection, Command Compilation, Command Execution, Telemetry Ingestion, and Activity Processing.

Splitting these into smaller, specialized interfaces—rather than larger, monolithic ones (e.g., a single Flight Dynamics interface)—simplifies development and testing. Although the hexagonal architecture is inherently monolithic, these interfaces can be implemented using adapters that interact with third-party systems. For instance, database adapters function as simple accessors. Additionally, Vyoma has implemented Orbit Determination as a Service, which is accessed via an HTTP client within the Orbit Determination Adapter (Figure 2).

Adapters can also implement multiple interfaces, allowing for mission-specific logic. Adapters are chosen based on configuration using a strategy pattern. They are selected based on configuration on constellation or satellite level by using the strategy and service-lookup-pattern [8].

Typically, a mission adapter includes:

- Multiple *Activity Processor* implementations.
- *Command Compilation* implementation for each platform.
- *Command Execution* implementation for each satellite or satellite group that shares a common communication protocol.
- *Optimization Constraint Provider* responsible for defining hard and soft constraints considered by the optimizer.
- *Telemetry Ingestors* for each platform, converting collected telemetry into a normalized domain model for subsequent processing. For example, GNSS telemetry may be prepared for Orbit Determination. These ingestors are also responsible for closing the planning request feedback loop. Once processes have been executed on the satellite and the respective telemetry has been received the status of a planning request can be updated.

3.3. Use Cases

Use cases, as per the hexagonal architecture, represent isolated blocks of logic which can be executed atomically, triggered by a manual or automatic trigger. It is an orchestration of domain logic using implementations of domain interfaces (i.e. adapters). According to this definition, a use case in this context **is not the same** as a use case in the usual meaning (hereinafter referred to as regular use case), i.e. the ways users or systems would use this system. A regular use case can be modelled by use cases combined with trivial interactions of users which eventually lead to the desired outcome.

Based on the requirements in Section 2, the following use cases have been identified: Planning Session Creation, Planning Session Scheduling, Planning Session Confirmation, Planning Session Publishing, Orbit Determination, Orbit Update, Orbit Distribution, and Telemetry Fetching. Since the system is designed to accommodate these predefined use cases, adding new ones is not anticipated. Instead, their behavior can be modified by configuring or replacing adapters. This is achieved through a combination

of the strategy pattern, dependency injection, and configuration [8]. Use cases can also trigger other use cases to enable high-level orchestration. For example, Telemetry Update automatically triggers Orbit Determination, ensuring seamless operational flow.

3.4. Main planning workflow via Activity Processors

The main planning workflow (Figure 3) consists of a sequence of use case executions. These use cases make use of the *Activity Processor* interface. This interfaces allows activities to define specific behavior along the path the planning workflow. Below is the pseudo-code defining the *Activity Processor* interface defining the main stages of the workflow which can be influenced by activities:

```
interface ActivityProcessor {

  int processingOrder();

  // Stage callbacks
  default void preProcess(planningSession, planningRequest) {};

  default Set<Process> createProcesses(planningSession, planningRequest, satellite) {
    return new Set();
  }

  default void postProcess(planningRequest) {};

  default void verify(planningSession) {};

  default void onConfirm(planningSession) {};

  default void onPublish(planningSession) {};

}
```

Every *Activity* can implement custom behavior for any stage. Only an implementation for `processingOrder` is required.

The `processingOrder` serves two critical purposes. First, it dictates the order in which different stages are invoked. This order must remain consistent within a single mission adapter (or constellation), ensuring that *Activity* implementations can reliably depend on it. A key example of this invariant is the guarantee that a `GroundStationContact.createProcesses()` is always invoked before `HealthCheckActivity.createProcesses()`, assuming that a health check can only happen during a ground station contact. This predefined sequence enables activities that must occur during ground station contacts to be explicitly linked to those contacts by constraining their scheduling within the relevant time frame.

The `preProcess` stage is responsible for modifying the initial conditions of planning. For example, a manoeuvring *Activity* might adjust the orbit so that later activities, such as a ground station contact Activity, can take this change into account when calculating contact opportunities.

The `createProcesses` stage generates all *Processes*, which serve as primary inputs to the scheduler. A process may include arbitrary parameters, which may be required by other use cases or stages, such as the optimization (by a *Constraint Provider*) or the *Telecommand Compilation* or *Execution*. Default parameters include whether the process is optional, has a fixed start time, or a fixed duration. Additionally, resource influences are attached to the *Process*, allowing the optimizer to estimate resource usage dynamically during execution.

The `postProcess` stage is invoked after scheduling and optimization, allowing for further refinements. For instance, it can support multi-level optimization. An example use case involves scheduling the *Process*

for a ground station contact *Activity*, followed by the attachment of additional *Processes* for other *Activities* which can only take place during a contact.

The `verify` stage ensures session validity by checking constraints that may not have been feasible or necessary to evaluate during solve time. E.g. using a high fidelity simulation tools provided by satellite manufacturers to ensure consistency and feasibility.

Finally, the `onPublish` is invoked after the planning session was promoted to operational. This stage can be used to e.g. modify resources after publishing. An example could be propellant mass which is a resource that only has to be tracked on the long term instead of having it to include as a constraint during optimization time unnecessarily difficulting the optimization problem.

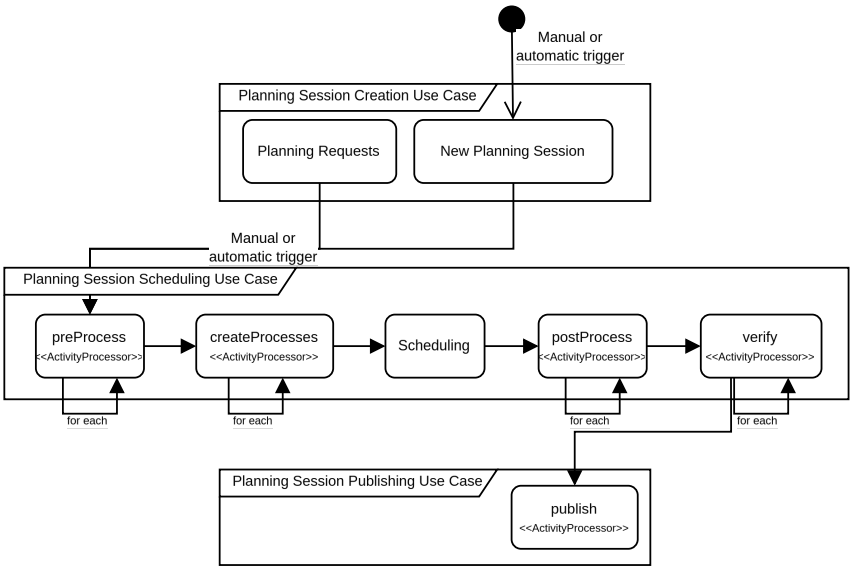

Figure 3: Workflow diagram showing the relationship between activity stages and the primary use cases.

4. Autonomous Mission and Satellite Operations

The standardized architecture, configuration, and domain model simplify support for autonomous operations. Automation within the system is primarily triggered by two mechanisms:

- **Time-based triggers**, defined by configurable schedules which can invoke use cases or domain service functionalities.
- **Event-based triggers**, such as:
  - The initiation or completion of use cases, enabling chaining of processes.
  - The storage or modification of objects (e.g., planning requests), which can trigger rescheduling for reactive and incremental planning.

4.1. Configurability and Operator Control

The system offers extensive configurability for autonomous operations, with all triggers managed through a standardized system-wide configuration approach. Throughout the mission lifetime, but crucially during early mission phases, operators retain the ability to manually invoke certain actions, including:

- (Re)-Invoking *Use Cases*
- Modifying the *Configuration*
- Adjusting *Resources* (e.g. capacity) or *Resource Influences*
- Creating or modifying *Planning Requests*

This offers the required balance between leveraging automation whenever its application delivers measurable benefits but foresees clear means of intervention for the operators in case of anomalies. However, once scheduling is completed, manual modifications to a planning session are restricted, except for the deletion of individual *Processes*. In such cases, the `ActivityProcessor.verify` stage is re-executed to ensure constraint validity and subsequently the default workflow is followed (Figure 3).

4.2. Core Automation Strategies

The foundation of configurable autonomous operations is built on the following strategies:

- **Planning Session Generation**: This strategy leads to the autonomous generation of planning sessions, triggering the actual planning iterations. The default strategy is a sliding window strategy creating a session each s hours with a duration of d hours and an overlap of l hours. These parameters can be configured per constellation. Also completely custom strategies can be implemented per constellation. For the overlapping parts of the sessions the strategy will try to synchronize the already scheduled processes in the previous session. This is done by identifying the planning requests that led to the processes and reapplying them to the new time-shifted session. Conflicts can happen if planning requests were created meanwhile. In this case a conflict resolution strategy is applied which also considers the `processingOrder` of an *Activity* (See Section 3.4).
- **Planning Request Generation**: Standing *Planning Requests* can be configured to be added automatically to every new *Planning Session* in order to conveniently accommodate recurring tasks. For each *Activity* a strategy can be configured. Examples strategies are: As often as possible, exactly n times or once as soon as possible. Typical examples include ground station contacts, power generation estimation and usually the primary mission goal. As *Activities* have custom implementations and support custom arguments this way arbitrary complex mission automation is possible.
- **Telemetry Ingestors**: Telemetry ingestors serve as triggers for automated actions after retrieving telemetry. Each ingestor monitors incoming telemetry for a given resource and can then invoke arbitrary logic. Usually an ingestor lives inside a custom adapter or a mission specific adapter. Ingestors are allowed to generate planning requests making them a powerful automation tool within the MPS. Examples of triggered actions are:
  - **Orbit Determination**: Triggered upon retrieval of GNSS telemetry.
  - **Reaction Wheel Saturation Estimation**: Can lead to the generation of a planning requests triggering a desaturation.
  - **Manoeuvre Estimation**: Continuously monitor propulsion subsystem usage to derive whether manoeuvres where successfully performed.

4.3. Observability, Monitoring, and Alerts

To enhance observability and introspection, an audit log system has been implemented. It follows a traditional software engineering logging approach but is enriched with metadata such as use case identifiers, satellite references, and constellation context. The audit log is also used for alerting. In cases where use cases or other components such as telemetry ingestors encounter an irregular or unexpected state, an error audit log entry is generated, and the use case terminates. If the use case is a session related one (e.g. session scheduling) the session will be moved into an error *state*. Error log entries trigger events in a monitoring and alerting system, which can be implemented via an adapter (e.g. for systems like Grafana). Operators then have to analyze the problem and decide for further action. Automatic problem resolution strategies can be implemented in certain cases but are usually adapter dependant and not embedded into the core MPS.

4.4. Data Distribution and Command Execution

The *Publishing Use Case* manages the distribution of all necessary mission data, including telecommands and operational orbital information. This process is usually platform-specific. Hence, adapters have to implement specific interfaces which convert from the domain model into the platform target representation and then also send it to the target systems.

It can be configured whether publishing of a session happens automatically after a session was scheduled and verified. In the initial stages of operations this can be turned off for manual verification.

By default, the publishing use case initiates the following downstream actions:

- **Compile and distribute Telecommands**: Managed via the GSN adapter.
- **Orbital Data Publication**: Shares orbit data with public catalogs such as the U.S. 18th Space Defense Squadron catalog (space-track.org).
- **ActivityProcessor On-Publish stage invocation:** Ensures each *ActivityProcessor* completes its finalization procedures.

5. Application to the Vyoma Mission

Vyoma's development of its MPS is, besides focusing on its flexibility and scalability as described in Section 3, driven by the complex Concept of Operations (ConOps) of the *Flamingo-1* mission. Next to the well-known and well-established use cases for a satellite mission, such as Earth Observation (EO) or communications satellites, operating a space-based telescope as part of a Space Situational Awareness (SSA) mission is a challenge. *Flamingo-1*'s ConOps requires the operations team on ground to find the optimal balance between observing and cataloging the space environment around Earth in *Surveillance* mode and performing dedicated tracking of individual objects, referred to as *Tracking* in the following. This needs to be accompanied by a contact schedule with the chosen ground station network that respects data and latency requirements.

Manoeuvres activities for orbit maintenance, as well as a rising number of collision avoidance manoeuvres (in the LEO regime) contribute to the complexity of the ConOps.

This section provides a concise overview of the implementation and configuration steps taken for *Flamingo-1*, in order to illustrate how an actual mission can be implemented using the presented architecture.

5.1. Flamingo-1 Mission Adapter

To accommodate the Vyoma Flamingo Mission within the new MPS, several mission-specific components have been implemented and configured. These include a dedicated mission adapter, a satellite-platform-specific telecommanding adapter, and an adapter for the communication with the chosen GSN provider next to adapters to Vyoma's own solutions for the FDS [9], Collision Avoidance (CA) and telemetry services. The mission adapter consists of a set of *Activity* implementations, the mission and spacecraft configurations together with a corresponding *ConstraintProvider* for scheduling constraints, along with resource tracking mechanisms, and configurations of the automation procedures.

The respective *Activity* definitions of each *Activity* account for buffers in the schedule such as ones required for changing the spacecraft's attitude or warming up its propulsion system, depending on the choice of satellite hardware. In some cases, buffers are added to single processes adjusting their start and end dates whereas in others, it proved to be beneficial to create multiple processes instead of only one. An example would be the *Manoeuvre Activity* of *Flamingo-1*, which requires a dedicated warmup phase of its thrusters. Modelling the manoeuvre using two processes, where the warmup process is automatically generated depending on the manoeuvre process, These exact definitions are accounted for during scheduling, resource propagation as well as during commanding, if required. However, the presented implementation features a two-stage approach to its resource propagation. During the optimization step itself, contributions to resource usages are considered to be the ones specified with the *Activity* definition

on an individual process level, whereas during the `verify` stage of each *Planning Session*, a high-fidelity model is configured to take into account the exact sequence of scheduled processes.

The following activities form the core of the *Flamingo* adapter: *Surveillance*, *Tracking*, ground station contact related activities, *Power Generation* and *Manoeuvre*. Tracking and Surveillance are the two payload operating modes, and therefore the primary mission modes. Tracking implements the `createProcesses` stage by running a space-based pass prediction algorithm to calculate observation opportunities. Together with the opportunity time frame the required attitude change is calculated and attached to the generated process. For *Flamingo-1*, the attitude is calculated upfront for each *Process* before running the optimization. This is realized via the choice for a baseline attitude that also matches the attitude of the default *Surveillance* operations.

Further, we identified two different types of activities related to ground station contacts. Firstly, the *Ground Station Contact Activity* itself that represents the time frame during which there is contact to a station. Secondly, activities that represent actual procedures that require to be executed during a contact. These include: Uplinking files (e.g. for upgrading on-board systems or the on-board schedule), health checks as well as downlinking of payload data and telemetry. The processes for this kind of activities are only added to the schedule after the schedule stage was invoked making it a multi level optimization. The processes are scheduled in order of their `Activity.processingOrder` (See Section 3.4).

The *Power Generation* activity is a special activity as it does not represent a real operational capability of the *Flamingo-1* spacecraft. However, as already introduced for the *Ground Station Contact Activity* as the bare concept of a contact window, its implementation demonstrates that any aspect of a spacecraft can be modelled using activities and the processes generated therefrom. As introduced in Section 3.1, the system supports modelling any form of *Resource Influence* and attaching it to an *Activity*. Therefore, no special representation or implementation is required for power generation in the domain model. For *Flamingo-1* the implementation of this mechanic was implemented via the corresponding process generation (activity stage `createProcesses`, see Section 3.4) which maps the predicted sun light exposure periods onto specific processes.

Activities which require activation or an action by the spacecraft (i.e. not the *Power Generation* activity) implement the commanding compilation and execution interfaces. They prepare all the parameters which are required by the platform-dependant commanding adapter based on the processes of a scheduled and verified planning session. Such an adapter was also implemented for Flamingo-1. It consumes these parameters, then synthesizes the entire onboard schedule by generating a program which is then sent to the C2 which is in charge of finally uplinking the commands. Also, it generates a set of telecommands which are not invoked onboard, but by the GSN. This adapter can be reused for further satellites with the same platform. also simplifies the telecommanding. Further, custom telecommands can now easily be mapped to the different processes using the *Command Compilation* and *Command Execution* interfaces.

Due to the nature of space missions, on-board resources are the constraining factor, especially in the range of small spacecrafts in LEO. This reduces the scheduling problem for *Flamingo-1* to the allocation of as much time as feasible to activities that contribute to the formulated mission goals, which are discretized in the form of opportunities (i.e. *Processes*), each assigned with a corresponding weight and their resource demands (i.e. *Resource Influence*). In order to formulate a manageable and solvable optimization problem, the set of tracked resources includes the following types of on-board resources, which are propagated and monitored during scheduling, and verified by higher fidelity models during verification of each *Planning Session*:

- Battery state of charge
- On-board data storage
- Reaction wheel speeds

5.2. Optimization

The scheduling problem at hand, as introduced in the subsection above, has a wide variety of solutions. However, in order to maximize the return of a space mission that is constrained by resource scarcity and a limited mission life time, employing an optimization step is imperative in being the second key factor next to the automation of the MPS.

Hence, the *Flamingo* mission, for the whole constellation, poses the problem of maximizing time spent in payload operations while, at the same time, maximizing resource utilization and minimizing the latency on delivering data products while staying within resource constraints at all times, in order to qualify as a feasible solution in the first place. An example for a successfully verified and therefore feasible and optimized operations schedule, which was configured to specifically maximize the time spent in *Surveillance* mode, can be found in Figure 4. In the user interface, each *Activity* is displayed in a swimlane of the Gantt chart. Additionally, resource utilization is shown as well. In order to be considered as a feasible solution, there is a set of scheduling constraints that need to be fulfilled at all times. For *Flamingo-1* specifically, this includes for example limits on battery and on-board data storage capacities.

The capabilities of the optimization, as planned for the launch and early operations of *Flamingo-1*, do not additionally optimize for attitude slewing. Instead, the modeling of mission procedures that require attitude changes accounts for the time and resources connected to each corresponding process. Specifically, the discretization of *Surveillance* and *Tracking* in specific opportunities, each already defined by a fixed start time, fixed duration, and predefined attitude sequence, abstracts away this complexity from the solving itself, reduces complexity and therefore computational effort significantly.

5.3. Automation

Additionally to the already described automation mechanisms, the system was extended with the following automated processes:

- **Orbit Acquisition Manoeuvres:** In order to reach the target orbit, orbit acquisition manoeuvres have to be planned for an extended period of time. An additional adapter was implemented which integrates a tool part of the Vyoma FDS capable of performing such calculations using Q-Law and convex optimization [9]. The adapter will trigger the manoeuvre calculation periodically with the latest know spacecraft state. It then converts the result of the optimization runs into planning requests which can span a long time range which exceeds multiple planning sessions. Each new planning session will then plan and execute a subset of requested manoeuvres. The adapter also keeps track of executed manoeuvres monitoring telemetry. In case of an incident (e.g. non successful execution of on of the manoeuvres) a recalculation is requested automatically and the schedule (i.e. currently existing manoeuvring planning requests) are overridden. This process is very closely monitored by operators, which can also request a recalculation manually if required.
- **Collision Avoidance Manoeuvres:** Similar to orbit acquisition manoeuvres support for collision avoidance manoeuvres was added by writing a new Collision Avoidance adapter. This adapter integrates the Vyoma Platform Collision Avoidance service which already has extensive screening and manoeuvre calculation and optimization capabilities. In case high risk conjunctions are identified a manoeuvre is calculated and ingested by the MPS automatically, again leading to a manoeuvring planning request and an alert to an operator.
- **Flamingo Planning Session Generation Strategy**: A default sliding window generation strategy will be used. Automatically orchestrates the creation of new planning sessions, e.g a new session covering 24 hours, where the following session will be created halfway through the execution of the previously active session.
- **Planning Request Generation Strategies**: Nominal operations of the *Flamingo* constellation operations feature the automated generation of *Planning Requests* for the following set of activities:
  - *Tracking* requests on top of manually created instances are received from Vyoma's own space object catalogue, with their priorities taking into account the time that has passed since the last record of an individual object or region.

- *Surveillance* opportunities are identified continuously, and based on certain heuristics that take into account the population of visible regions and considering illumination conditions, are added to the schedule automatically.
- *Ground Station Contacts* are requested via applying a given set of constraints on overall contact time, or the distribution of contacts automatically and subsequently booked with the GSN provider.

5.4. Constellation Expansion

The Vyoma Flamingo mission will require up to 12 satellites in its constellation. For subsequent satellites to be incorporated after Flamingo-1 the required changes to configuration and (mission specific) implementation are limited. Mostly configuration related to the payload and telecommand are required. The implementation of the activities will be able to be reused. Especially the *Tracking* and *Surveillance* activities as they are mission specific activities rather than spacecraft specific ones.

6. Conclusion

This paper presents the design and implementation of a flexible and extensible Mission Planning System (MPS), and its application to a space-based space surveillance and tracking mission. The architecture emphasizes modularity, scalability, and autonomy, making it well-suited for managing heterogeneous satellite constellations with complex operational requirements. By employing principles from clean and hexagonal architecture, and leveraging a domain-driven design, the system enables seamless integration of diverse mission-specific components while ensuring maintainability and adaptability.

The application of the MPS to Vyoma's Flamingo-1 mission demonstrates how operational complexity can be effectively managed through a well-defined domain model, standardized interfaces, and a highly configurable automation framework. Key mission functionalities such as observation scheduling, ground station coordination, and manoeuvre planning are orchestrated through activity processors and optimized scheduling strategies. Additionally, the system supports incremental and reactive planning, robust telemetry ingestion, and secure command execution pipelines.

The MPS architecture not only streamlines mission integration and operational workflows but also reduces development time and operator workload. Its extensibility and abstraction capabilities position it as a viable platform for future space missions across various domains, contributing to the advancement of autonomous satellite operations and space situational awareness.

7. Outlook

Until the launch of the Flamingo-1 satellite, the system will undergo further improvements and rigorous testing against the Flamingo mission operations concept. A key focus will be evaluating whether the domain model and interfaces are sufficiently flexible and complete. Additionally, the second generation of satellites of the *Flamingo* constellation, *Flamingo-2*, is currently under development. This will add another platform and the next iteration of the payload to the *Flamingo* constellation, and therefore the constellation operated by the MPS. This will require integration and testing of this platform, particularly the different the telemetry and telecommanding (TMTC) functionalities as well as payload commanding.

Further enhancements are also planned to be made to the scheduling related components, leveraging different optimization algorithms to further improve tracking and surveillance efficiency. In addition to these two payload modes, the target is to implement a combined operational approach, employing a sweeping surveillance that covers a greater portion of the observable environment with each process and therefore increases the return per scheduled instance.

While the system already supports both incremental and from-scratch planning, neither incremental nor reactive planning have yet been finally evaluated. This aspect will be systematically analyzed to determine their applicability and effectiveness.

We anticipate that development and integration times for new missions will be significantly reduced. However, this assumption must be validated. Verification will take place within the framework of the Autonomous Satellite Operation (AUTOPS) joint research project as well as with pilot customers. Moreover, we aim to assess and quantify the impact of the system's design and automated procedures on human operator workload.

Finally, large-scale constellation testing will be conducted, evaluating system performance for fleets ranging from tens to hundreds of satellites.

8. Appendix

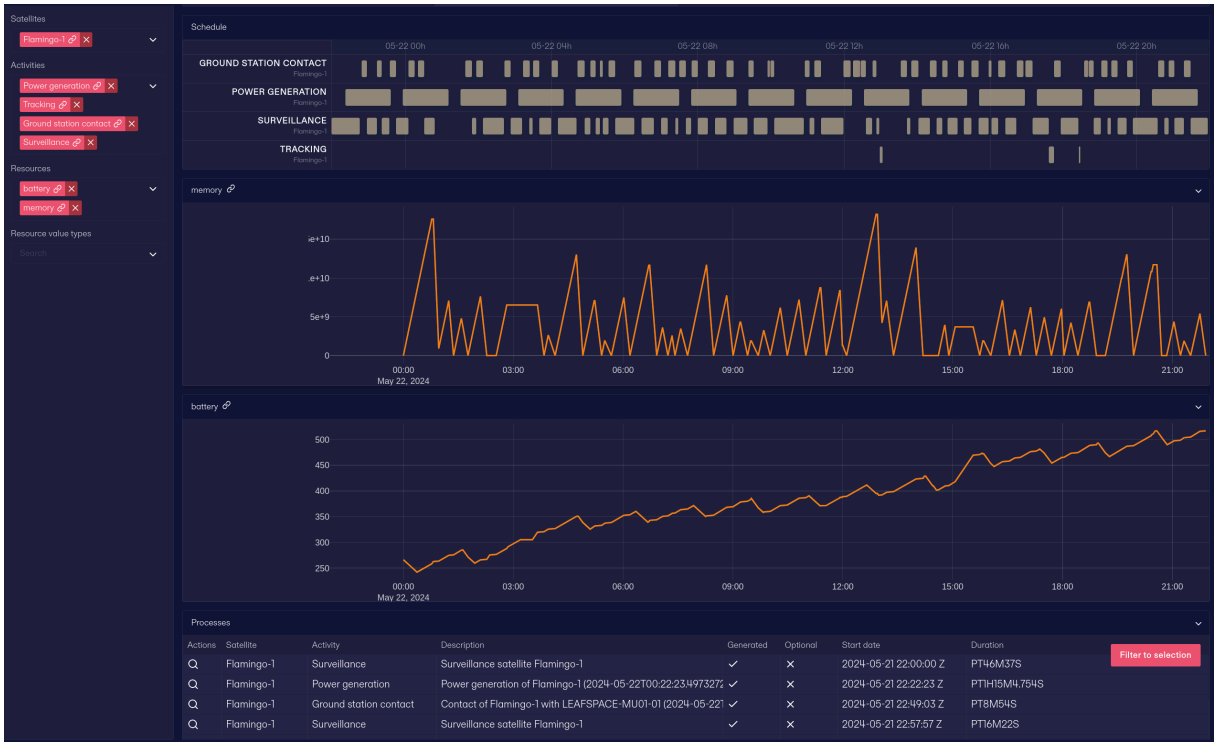

Figure 4: Screenshot of the MPS frontend showing an optimized schedule for the Flamingo-1 satellite.

Acknowledgements

This work is supported by the Joint Project *Autonomous Satellite Operation* (AUTOPS) within the funding line "Mobility" (Space sector) in accordance with the guidelines of the Bavarian Collaborative Research Program of the StMWi (BayVFP; Funding Reference Number: MRF-2307-0004) with the consortium partners Vyoma GmbH (Consortium Leader), Technical University of Munich (TUM) – Chair of Spacecraft Systems, and the German Aerospace Center (DLR) – Mission Operations (RB-MIB).

Bibliography

[1] M. T. Wörle *et al.*, "The Incremental Planning System GSOC's Next Generation Mission Planning Framework," in *SpaceOps 2014 Conference*, 2014, p. 1785.

[2] Consultative Committee for Space Data Systems (CCSDS), *Orbit Data Messages. Blue Book.*, CCSDS502.0-B-3 ed. 2023.

[3] C. Iacopino, S. Harrison, and A. Brewer, "Mission Planning Systems for Commercial Small-Sat Earth Observation Constellations," 2015.

[4] E. Maurer *et al.*, "TerraSAR-X mission planning system: Automated command generation for spacecraft operations," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 48, no. 2, pp. 642–648, 2009.

[5] R. C. Martin, "Clean architecture." Prentice Hall, 2017.

[6] S. Freeman and N. Pryce, *Growing Object-Oriented Software, Guided by Tests*, 1st ed. Addison-Wesley Professional, 2009.

[7] E. Evans, *Domain-driven design: tackling complexity in the heart of software.* Addison-Wesley Professional, 2004.

[8] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software.* Pearson Deutschland GmbH, 1995, p. 315.

[9] M. Trisolini *et al.*, "Flight Dynamics System for Space-Based Space Surveillance and Tracking," in *SpaceOps 2025*, Montreal, Canada, May 2025.