Thematic Summary of the Focus Group with Aerospace Domain Experts

1. Motivation and Context

The team clarified that the proposed architecture stems from a need to support the design and reconfiguration of scheduling configurations in a more structured and automated way. Traditionally, this process is manual, slow, and error-prone, especially when adapting to changing business missions or third-party requests. The goal is to enable a semi-automated decision support system grounded in a Digital Twin.

2. Digital Twin vs. Simulator

Participants emphasized that the Digital Twin is not merely a simulator. While a simulator is static and passive, the Digital Twin is proactive, continuously integrating data from the Physical Twin to refine its internal model. This makes it possible to evaluate and propose changes aligned with real-time satellite behavior, especially useful in evolving mission scenarios.

3. Scheduler Design and Optimization

The static schedule configuration is driven by a graphical tool that ensures structural conformance through a metamodel. It encodes task properties (offsets, priorities, dependencies) and serves as input to an optimizer. The optimizer applies a two-level approach: linear/quadratic programming for priority constraints and differential evolution for offset tuning. The goal is to reduce preemptions, respect timing constraints, and handle functional dependencies between tasks.

4. Simulator Capabilities

The simulator, built in-house, is an event-driven engine with configurable resolution (e.g., 0.1 ms). It validates the generated schedules by executing task traces, detecting preemptions, and visualizing idle periods. While it does not simulate low-level hardware behavior, it can be configured to account for bus usage or DMA constraints, offering a realistic approximation of system execution under different scheduling configurations.

5. Runtime Adaptation and Validation Loop

Any newly generated configuration must be validated by an expert before being uploaded to the satellite. Once deployed, the actual execution is monitored. If telemetry reveals drift or anomalies (e.g., excessive preemptions, underutilization, missed deadlines), the loop restarts. This continuous cycle constitutes the runtime adaptation mechanism, closing the Digital Twin feedback loop.

6. Certification and Safety Concerns

Due to safety and qualification constraints, automatic application of changes is not allowed. A supervisor must always approve new schedules before they are sent to the satellite. The system is positioned as a decision support tool, not as a fully autonomous controller, to remain compliant with safety-critical regulations.

7. Data Interpretation and Wall Time Analysis

There was discussion on measuring actual execution times (CPU time vs. wall time) to reconstruct execution traces. While the simulator currently works with assumed CPU times, there’s ongoing work to improve accuracy using observed telemetry. This could support failure analysis, bottleneck detection, and system tuning.

8. Graphical Tool and Model-Based Design

The GUI was designed as a structured alternative to the traditional Excel-based process. Tasks are defined with strict conformity to a metamodel, enabling automated checks and transformations. It ensures syntactic and semantic validity and acts as a frontend for both the simulator and optimizer.

9. Real Use Case Scenarios

A recurring example involved reusing satellites initially launched for a specific mission (e.g., ESA), and later repurposing them by renting onboard resources to new clients (e.g., TIM, Leonardo). This requires inserting new task sets not foreseen at launch, making runtime rescheduling necessary.

10. Open Challenges and Future Work

Participants acknowledged areas for improvement: managing bus saturation, modeling data aging (especially for sensors), and better integration of data-dependency constraints. Future extensions of the toolchain may include more dynamic constraint handling, relative fences, and predictive anomaly detection using telemetry.