

Operation Plan Management System (OPMS): Software for the Safe Operations of Satellites

By Shunichiro NOMURA,¹⁾ Tomoki MOCHIZUKI,¹⁾ Kota KAKIHARA,²⁾ Yosuke KAWABATA,¹⁾
Hirotaka KONDO,¹⁾ Kentaro ENOKIDA,¹⁾ Akihiro ISHIKAWA,¹⁾ Shuhei MATSUSHITA,^{1),2)}
Ryu FUNASE,¹⁾ and Shinichi NAKASUKA¹⁾

¹⁾The University of Tokyo, Tokyo, Japan

²⁾ArkEdge Space Inc., Tokyo, Japan

This paper presents the Operation Plan Management System (OPMS), a web-based software for the safe operations of satellites. The OPMS controls an operation plan created by its users with a hierarchical structure of the operation plan and a concept of action and action plan to describe the sequence of operations. It then validates the operation plan using the simulator and manages the validation results by modeling the action plan as a finite state machine. Finally, it synchronizes the validated operation plan with the satellite and manages the synchronization results with three versions of the operation plan. Its implementation and demonstration in SPHERE-1 EYE satellite are also presented.

Keywords: Satellite Operation, Software, Ground System

Nomenclature

$\mathcal{A}$	: set of all the action plans
$\mathcal{A}_S$	: set of all the action plans in the synchronized operation plan
$\mathcal{A}_{UV}$	: set of all the action plans in the unvalidated operation plan
$\mathcal{A}_V$	: set of all the action plans in the validated operation plan
c	: action plan cancellation state
e	: action plan execution state
ETNR	: execution time not reached state
ESOS	: execution started on satellite state
ETRBS	: execution time reached before sync state
r	: action plan registration state
s	: action plan state
s	: synced state
$\mathcal{S}$	: set of all the action plan states
ur	: unrequested state
uv	: unvalidated state
v	: validated state

1. Introduction

1.1. Background

The process of operation planning is a crucial aspect of satellite management. It involves defining the objectives of the operation plan, creating a draft of the operation plan in the form of a command file, and reviewing the operation plan by the operation team. The reviewed command files are executed during the operation and the commands are sent to the satellite. However, this process is inefficient for several reasons.

One of the primary reasons for its ineffectiveness is the difficulty in sharing the operation plan among the subsystem teams without a single source of truth. This can lead to confusion and errors during the execution of the operation plan. Additionally, the operation team has to review the operation plan manually, which can be time-consuming and prone to errors. Moreover, the operation team has to execute the command files manually, which can result in delays and inefficiencies.

1.2. Contributions

To solve these problems, we developed a web-based software called the Operation Plan Management System (OPMS). OPMS is designed to allow the subsystem teams to create the operation plan and check its validity automatically. This software ensures that all subsystem teams have access to the same information, thus eliminating confusion and errors. Additionally, OPMS can be connected to the ground station software, so the operation plan can be sent to the satellite automatically, reducing delays and improving efficiency.

The OPMS software provides a solution to the inefficiencies in the operation planning process. It enables the subsystem teams to collaborate effectively, thereby reducing errors, ensuring efficient execution of the operation plan, and improving overall productivity.

2. Related Works

The field of satellite operation plan management software has several notable contributions. Tools for mission planning and scheduling come in various forms and capabilities, offering different solutions for managing satellite operations.

One prominent example is the planning framework by the German Aerospace Center (DLR) German space operations centre (GSOC), which has been successfully applied to various space missions including crewed missions such as Spacelab, D-2, X-SAR/SRTM, MIR, and

uncrewed scientific missions such as CHAMP, GRACE 1&2, TerraSAR-X and TanDEM-X, FireBird, Eu:CROPIS, Grace-FO, EnMAP, and commercial missions such as TDP-1, EDRS-A&-C, PAZ and SmallGEO HAG-1^{1,2)}. The suite includes the Reactive Planning Framework, Plains, PintaOnWeb, SCOTA, SPOT, PLATO, PINTA, SoEEditor, and TimOnWeb, each serving unique functionalities and capabilities. They collectively offer a broad range of mission requirements, including collecting and analyzing information, resources, and requirements; planning onboard and related on-ground activities; generating the mission timeline; and producing and distributing necessary outputs. Moreover, they are developing the capability in the onboard planning autonomy, including the VAMOS experiment on the BIROS satellite³⁾.

Another example is the Operation Planning Tool for Earth-observation MISSION (OPTEMIS) from Sentinel Asia⁴⁾. This tool has four main modules: user interface and data management; feasibility assessment; planning and scheduling; and satellite simulator. Based on the use input, it creates and optimizes the mission plan while checking the feasibility of the plan using the satellite simulator.

STM’s Mission Planning Software is another solution in the field⁵⁾. This web-based application offers automatic planning without user intervention, feasibility analysis, satellite power and memory management, and support for different imaging request types. It is designed to plan operations for missions consisting of one or more satellites, subject to constraints and resources.

The other mission planning tools include the OrbitLogic’s Collection Planning and Analysis Workstation and STK Scheduler^{6,7)}, CGI’s Satellite Resource Planning Software⁸⁾, TERMA’s CCS5⁹⁾, Lockheed Martin’s Horizon™ Command & Control (C2)¹⁰⁾ and Compass™ Mission Planning software¹¹⁾, Optimyz™ from Parsons¹²⁾, and the Mission Operations Planning and Scheduling System (MOPSS) currently used at the Goddard Space Flight Center (GSFC)¹³⁾.

3. OPMS Overview

3.1. Control function

3.1.1. Operation plan hierarchal structure

The Control function of the Operation Plan Management System (OPMS) software provides an essential service by managing the operation plan data with structured models. These structured models, as shown in Fig 1, divide an operation plan along the time dimension into multiple sub-operation plans. For instance, a sub-operation plan may correspond to one orbital revolution, and this division enables more granular control over the operation plan.

An aggregate unit of operation is represented as an action plan, which includes tasks such as preparing for a pass or initializing an instrument for observation. The action plans are designed to be modular and easily modifiable, which allows for greater flexibility in managing operations.

A sub-operation plan contains timeline information of action plans and passes within it, which makes it an essential tool for managing complex operations. This feature enables the operation team to track the progress of the operation plan in real-time and make necessary adjustments to ensure the smooth execution of the operation.

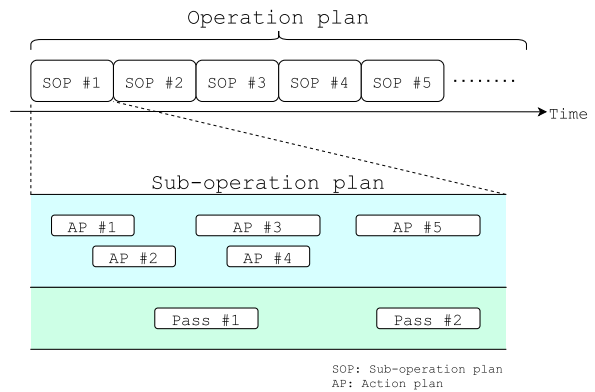


Fig 1. Hierarchical structure of an operation plan.

3.1.2. Action and action plan

Within the Operation Plan Management System (OPMS), an action is considered the fundamental unit of operation planning. It is a parameterized template for an action plan and represents an aggregate operation procedure. An action contains a sequence of commands and the relative time at which each command is executed, as well as a list of parameters to hydrate some command arguments when creating the action plan. For example, the start_realtime_imaging action in SPHERE-1 EYE mission consists of two commands, as shown in Table 1. The second argument of MISSIONIF_CONFIG command and the first argument of START_REALTIME_STREAMING command are parameterized, as shown in Table 2.

When creating an action plan, the user must specify the action, the action plan start time, and the argument to the action parameters, as the example shown in Fig. 2. In this example, the action consists of three commands: Command_A, Command_B, and Command_C. The first command is executed at the start time of the action plan, the second one is executed 3 seconds after the start time, and the third one is executed 8 seconds after the start time. The first arguments of Command_A and Command_C (x and y, respectively) are parameterized, and the user must specify the values of the parameters and the start time of the action plan when creating the action plan.

The benefits are particularly noticeable when the satellite bus developers and satellite operators work as separate teams. By defining the actions, the satellite bus developers can constrain the sequences of commands that can be executed on the satellite and prevent undesired, risky operations, and the satellite operators can be confident in the operation plans that are pre-defined in coordination with the satellite bus operator while having the flexibility of the execution time of the action plans and the parameterization of the action.

It is important to note that while users are able to create or delete an action plan, they are not able to modify it. This ensures that the action plan remains consistent and unaltered throughout the execution process.

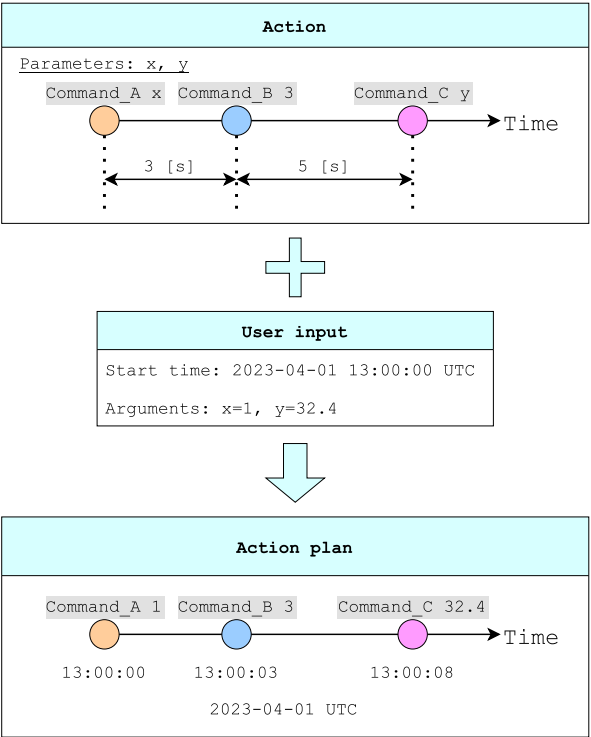


Fig. 2. Action plan creation process.

Table 1. List of commands for the start_realtime_imaging action.

Δt [s]	Command	Arg 0	Arg 1
0	MISSIONIF_CONFIG	0xFFFFFFFF	Param0
10	START_REALTIME_STREAMING	Param1	N/A

Table 2. List of parameters for the start_realtime_imaging action.

Index	Data type	Description
0	uint8	Specifies CCC rate
1	uint32	Bitrate [kbps]

3.2. Validate function

During the validation phase of the Operation Plan Management System (OPMS), determining whether a given sub-operation plan can be safely executed on the satellite is a critical concern. However, this process is complicated by several challenges that need to be addressed. Firstly, it depends on the state of the satellite at the beginning of the sub-operation plan (SOP), which cannot be known at the time of the validation. This is because the satellite’s state is continuously changing, and it is impossible to predict its exact state at any given time.

Secondly, the state of the satellite can be changed by inert uncertainty and modification of other SOPs before the one to validate. This means that if some prior SOP (denoted

as SOP A) becomes more aggressive, e.g., more energy-consuming while stays within the operation constraints, an SOP (denoted as SOP B) after SOP A can be invalid because the initial state of SOP B becomes more severe. This presents a significant challenge when validating the sub-operation plan because it is difficult to determine the precise state of the satellite when the SOP starts.

To overcome these challenges, the OPMS has developed a solution that involves fixing the initial and terminal conditions and sharing them with the adjacent SOPs. The SOP is considered OK if, starting from the fixed initial condition, the terminal state is not worse than the given fixed terminal condition. An example of validation with shared boundary conditions of the battery depth-of-discharge is shown in Fig. 3. This conservative assessment ensures that the sub-operation plan can be safely executed on the satellite and its validity is not affected by other SOPs prior to the SOP in question.

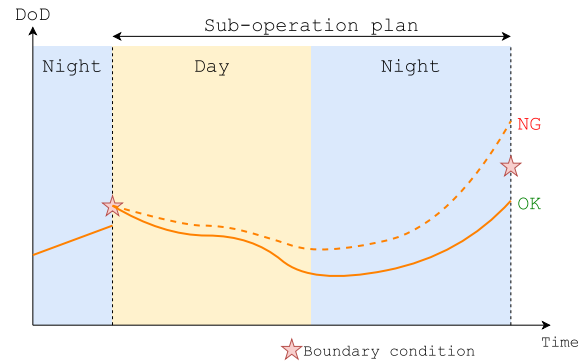


Fig. 3. Validation with the shared initial and terminal conditions.

3.3. Synchronize function

3.3.1. Three conceptual versions of the operation plan

The process of synchronizing the validated operation plan with the satellite presents several challenges that must be addressed. One of the most significant challenges is keeping track of what has been validated and what has not. Without a clear understanding of what has been validated, it can be difficult to ensure that the operation plan executed on the satellite is validated. To address this issue, the OPMS has developed three conceptual versions of the (sub-)operation plan: Unvalidated, Validated, and Synchronized.

The unvalidated version of the operation plan is like a draft of the operation plan. It is created and modified directly by the user. When the user requests a validation, the unvalidated version is simulated for validation.

The validated version is a snapshot of the unvalidated version at the latest successful validation. This version is created after the user has submitted the unvalidated version for validation and has received an OK result. Once the validated version is created, it cannot be modified unless there is a newer successful validation. It represents the latest validated operation plan, and is synchronized with the satellite during the synchronization step. This ensures that the operation plan on the satellite is always validated.

Finally, the synchronized version is a snapshot of the validated version at the latest synchronization. This version is created after the validated version has been synchronized with the satellite. The purpose of the synchronized version is to keep track of what is already on the satellite and what is not. This information is used to register or cancel action plans during the synchronization step.

To illustrate how this system works, consider the following example steps shown in Fig. 4: (1) create and modify the unvalidated SOP 1&2, (2) validate SOP 1 and receive an OK result, freezing it into the validated SOP 1, (3) validate SOP 2 and receive an NG result, leaving the validated SOP 2 empty, and (4) synchronize the operation plan, where only the validated SOP 1 is synchronized with the satellite, and copied to the synchronized SOP 1

By using these conceptual versions, the OPMS ensures that the operation plan that is executed by the satellite is the same as the validated version. These conceptual versions are managed using a state machine where each action plan has its own state and the state is transitioned by events such as user input and operation plan synchronization with the satellite. The details of the state machine is described in Section 3.3.2.

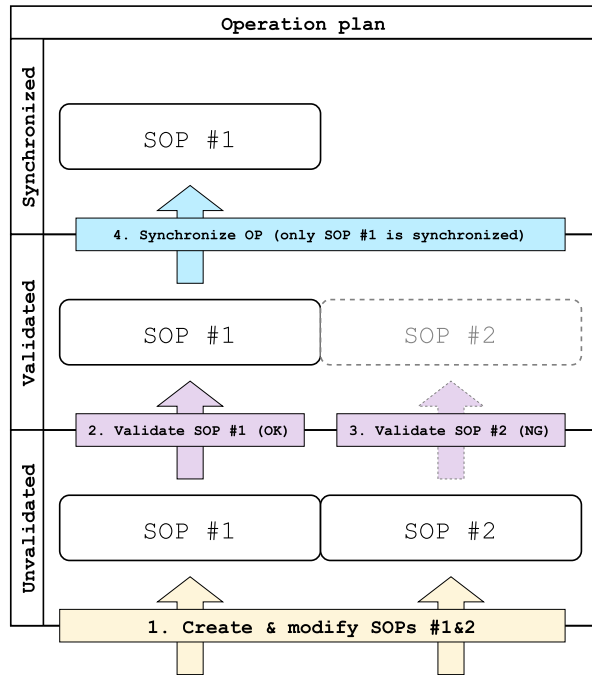


Fig. 4. Three conceptual versions of the (sub-)operation plan.

3.3.2. Action plan state machine

To keep track of the three versions of the sub-operation plan, an action plan is modeled as a state machine where the state is defined as

$$s = (r, c, e) \quad (1)$$

where

$$\begin{aligned} r &\in \{uv \text{ (unvalidated)}, v \text{ (validated)}, s \text{ (syncd)}\} \\ c &\in \{ur \text{ (unrequested)}, uv \text{ (unvalidated)}, \\ &\quad v \text{ (validated)}, s \text{ (syncd)}\} \\ e &\in \{ETNR \text{ (execution time not reached)}, \\ &\quad ESOS \text{ (execution started on satellite)}, \\ &\quad ETRBS \text{ (execution time reached before sync)}\}. \end{aligned} \quad (2)$$

r is the registration state that represents whether the addition of the action plan is validated and/or synchronized. When an action plan is first created by the user, it is in the unvalidated state uv . When the sub-operation plan containing the action plan is validated, the action plan is moved to the validated state v . When the validated sub-operation plan is synchronized with the satellite, the action plan is moved to the synced state s .

c is the cancellation state that represents whether the removal of the action plan is requested, validated and/or synchronized. When an action plan is first created by the user, it is in the unrequested state ur . When the user requests the removal of the action plan, it is moved to the unvalidated state uv . When the sub-operation plan containing the deleted action plan is validated, the action plan is moved to the validated state v . When the validated sub-operation plan is synchronized with the satellite, the action plan is moved to the synced state s .

e is the execution state. When the execution time of the action plan is not reached, it is in the “execution time not reached” state $ETNR$. When the execution time of the action plan is reached before the sub-operation plan containing the action plan is synchronized with the satellite, it is in the “execution time reached before sync” state $ETRBS$. When the execution of the action plan is started on the satellite, it is in the “execution started on satellite” state $ESOS$.

Fig. 5 shows the state diagram of the action plan. A state transition is triggered by one of the following events:

- **add**: the action plan is created by the user.
- **validate**: a validation of the sub-operation plan containing the action plan is requested by the user. The next state is determined by the result of the validation.
- **sync**: the sub-operation plan containing the action plan is synchronized with the satellite.
- **delete**: the user requests the removal of the action plan.
- **ExecutionTimeReached**: the execution time of the action plan is reached.

As observed in Fig. 5, the registration state r and the cancellation state c can be ignored when the execution state e is either $ESOS$ or $ETRBS$. For this reason, we define the aggregated states $ESOS$ and $ETRBS$ as follows:

$$\begin{aligned} ESOS &:= \{(r, c, e) \in \mathcal{S} \mid e = ESOS\} \\ ETRBS &:= \{(r, c, e) \in \mathcal{S} \mid e = ETRBS\} \end{aligned} \quad (3)$$

where $\mathcal{S}$ is the set of all states. The notations $ESOS$ and $ETRBS$ can refer to either the aggregated states or the execution states, depending on the context.

Using the action plan state, the three conceptual versions of the operation plan can be defined as follows:

$$\mathcal{A}_{\text{UV}} := \{(r, c, e) \in \mathcal{A} \mid c = \text{ur}\}$$

$$\mathcal{A}_V := \{(r, c, e) \in \mathcal{A} \mid r \in \{v, s\} \wedge c \in \{ur, uv\}\} (4)$$

$$\mathcal{A}_S := \{(r, c, e) \in \mathcal{A} \mid r = s \wedge c \in \{\text{ur}, \text{uv}, \text{v}\}\}$$

where $\mathcal{A}_{UV}$, $\mathcal{A}_V$, and $\mathcal{A}_S$ are the sets of the action plans included in the unvalidated, validated, and synced operation plans, respectively, and $\mathcal{A}$ is the set of all the action plans.

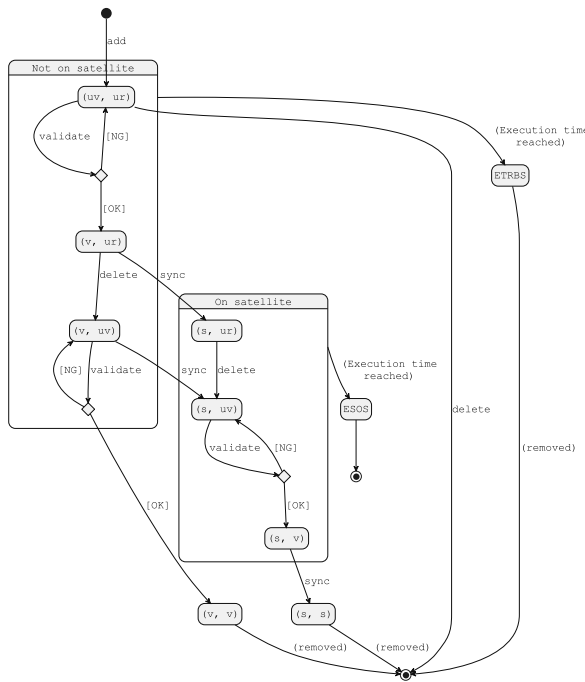


Fig. 5. Action plan state diagram. All the states whose execution state e is either ES0S or ETRBS are represented as aggregated states ES0S and ETRBS, respectively. For the other states, whose execution state e is ETNR, the state is represented as a tuple (r, c) where r is the registration state and c is the cancellation state. ur, uv, v, and s are the unrequested, unvalidated, validated, and synced states, respectively. The labels of the transitions add, validate, delete, sync, Execution time reached are the events that trigger the transition. removed indicate that the action plan is removed from the state machine and no longer exists.

3.4. Implementation

The OPMS is implemented as a web application using C# and .NET Core. The web application is hosted on Amazon Elastic Container Service (ECS) and uses AWS Aurora MySQL as the database. The web application exposes a RESTful API that is used by the user to create and delete the operation plan, and to request the validation and synchronization of the operation plan.

The validation of the operation plan is run by the OPMS via a job runner middleware, and the validation job is executed using the S2E¹⁷⁾ and Python scripts. It simulates AOCS, power, thermal, and communication subsystems and the simulation result is checked against the operation constraints such as tracker-to-sun angle, battery depth of discharge, temperature constraints, and communication link margin.

4. In-orbit demonstration

The OPMS has been deployed and being used for SPHERE-1 EYE, an Earth observing satellite developed jointly by Sony Group Corporation, JAXA, and the University of Tokyo, and launched on January 3, 2023. After the launch, the three core functions of the OPMS (*Control*, *Validate*, and *Synchronize*) were demonstrated against the satellite in orbit.

4.1. Control function

In SPHERE-1 EYE, a sub-operation plan corresponds to one revolution around the Earth. As of May 23 2023, the satellite bus operator and the user have defined 132 actions as templates for operation procedures, and more than 1,900 sub-operation plans and more than 140 action plans have been created in the OPMS. Thanks to the OPMS hierarchical data structure, these sub-operation plans and action plans have been created and modified successfully via the API exposed to the user.

4.2. *Validate* function

Since the launch of SPHERE-1 EYE, 77 sub-operation plan validations have been requested by the user, the sub-operation plans been simulated, and operation constraints been checked. However, due to the unexpected operation mode in orbit and insufficient in-orbit data in the mode, there is a discrepancy between the assumption of the simulation and the actual operation. The accuracy of the simulation should be verified and improved as we collect more in-orbit data.

4.3. Synchronize function

Since the launch of SPHERE-1 EYE, 149 action plans have been successfully registered and executed on the satellite.

One of the challenges in synchronizing the operation plan is interruption of the command transmission sequence. The uplink to the satellite can be unstable depending on the satellite attitude and the antenna pattern, and it's possible that some of the commands transmitted from the ground stations cannot be received by the satellite. This can be critical if, for example, only the power-on command of some component is registered on the satellite and the power-off command is interrupted and not received, in which case the component will not be powered off, resulting in unexpected power consumption.

In SPHERE-1 EYE, we took the two strategies to tackle the problem: a) use of Type-AD commands and b) atomic synchronization. Similar strategies should be taken when applying to other satellites.

A Type-AD command is a command transmitted to the satellite using TC transfer frames of Type-AD, ensuring that the commands are executed on the satellite in the same order of transmission, and if the transmission is interrupted, all the subsequent commands will be rejected until the operator retransmit from the interrupted command¹⁴⁾. The OPMS always sends commands as Type-AD commands and avoids command interruption.

However, sending Type-AD commands still cannot prevent the partial synchronization problem. It is still pos-

sible that the command transmission is interrupted *after* registering the power-on command and all the subsequent commands, including the power-off command, are rejected. To prevent such partial synchronization, we designed an atomic synchronization sequence using C2A's block commands^{15,16)}. A block command in C2A is a sequence of commands grouped together that can be executed as if it were a single command. Each command in a block command has an execution time attribute that defines the time at which the command is executed relative to the block command execution (called "deployment" in the C2A's terminology). One of the functionalities of the block command is that it can be activated and deactivated, enabling the atomic synchronization. The OPMS registers a new sub-operation plan to a satellite using a main block command and sub block commands in the following steps:

1. Clear the existing commands in the main block command. The block command is by default inactive after being cleared.
2. Schedule the deployment of the main block command at the start time of the sub-operation plan.
3. Register a sub block command deployment to the main block command for each action plan.
4. Register the commands of each action plan to the sub block command.
5. Activate the sub block command.
6. Activate the main block command.

This procedure is atomic because no action plan in the sub-operation plan will be executed without the activation of the main block command in Step 6.

5. Conclusions

In this paper, we presented the design and implementation of the OPMS, an operation plan management system with three core functions: *Control*, *Validate*, and *Synchronize*. An operation plan has a hierarchical data structure consisting of sub-operation plans and action plans, and the OPMS exposes a RESTful API to the user to create and delete the operation plan, and to request the validation and synchronization of the operation plan. A sub-operation plan is validated upon the request of the user, and the sub-operation plan is simulated and operation constraints are checked against the boundary conditions, allowing the independent validation of each sub-operation plan. The OPMS synchronizes the operation plan by registering the commands of the action plans to the satellite in the order of execution, and the synchronization is atomic to prevent partial synchronization. The OPMS has been deployed and being used for SPHERE-1 EYE, an Earth observing satellite developed jointly by Sony Group Corporation, JAXA, and the University of Tokyo, since its launch on January 3, 2023, and the three core functions of the OPMS have been demonstrated against the satellite in orbit.

Acknowledgements

The authors would like to thank all the project members for their contributions to the development of SPHERE-1 EYE.

References

- 1) DLR: Software applications for mission planning, https://www.dlr.de/rb/en/desktopdefault.aspx/tabid-6816/4256_read-6303/ (accessed May 29, 2023).
- 2) DLR: Mission Planning Services, 2022-04-12_RB_SP_MissionPlanning_v02, https://www.dlr.de/rb/en/PortalData/38/Resources/dokumente/leistungen/DLR_RB_Portfolio_MissionPlanningServices.pdf
- 3) B. Wille, M. T. Wörle, and C. Lenzen: VAMOS – Verification of Autonomous Mission Planning On-board a Spacecraft, IFAC Proceedings Volumes, vol. 46, no. 19, pp. 382–387, Jan. 2013.
- 4) W. Vongsantivanich and S. Chaimatanan: The Development of Mission Planning Tool for Thailand's Earth Observation Mission, in SpaceOps Conferences, Daejeon, Korea, 2016.
- 5) STM Mission Planning Software, <https://satsearch.co/products/stm-mission-planning-software> (accessed Apr. 25, 2023).
- 6) Orbit Logic CPAW, Collection Planning and Analysis Workstation, <https://www.orbitlogic.com/collection-planning-and-analysis-workstation.html> (accessed May 29, 2023).
- 7) Orbit Logic STK Scheduler, <http://www.orbitlogic.com/stk-scheduler.html> (accessed May 29, 2023).
- 8) CGI Satellite Resource Planning Software, <https://www.cgi.com/uk/en-gb/brochure/space/cgi-satellite-resource-planning-software> (accessed May 29, 2023).
- 9) Satellite Control Systems, Terma. <https://www.terma.com/products/space/satellite-control-systems/> (accessed May 29, 2023).
- 10) Lockheed Martin: Horizon™ Command & Control, https://lockheedmartin.com/content/dam/lockheed-martin/space/documents/horizon-and-compass-product-sheets/23-02125_Horizon_MissionPlanning_Paper.pdf (accessed May 29, 2023).
- 11) Lockheed Martin: Compass™ Mission Planning, https://lockheedmartin.com/content/dam/lockheed-martin/space/documents/horizon-and-compass-product-sheets/23-02125_Compass_MissionPlanning_Paper_v2.pdf (accessed May 29, 2023).
- 12) Parsons Corporation: Automated Scheduling For Satellite Communications, Jan. 13, 2022. <https://www.parsons.com/products/optimiz/> (accessed May 29, 2023).

- 13) Goddard Space Flight Center: Mission Operations Planning and Scheduling System (MOPSS), NASA Tech Briefs. Sep. 2011. [Online]. Available: <https://ntrs.nasa.gov/api/citations/20120000449/downloads/20120000449.pdf>
- 14) CCSDS: Recommended Standard for Communications Operation Procedure-1, CCSDS 232.1-B-2, Sep. 2010. [Online]. Available: <https://public.ccsds.org/Pubs/232x1b2e2c1.pdf>
- 15) S. Nakajima et al.: Command-centric architecture (C2A): Satellite software architecture with a flexible reconfiguration capability, Acta Astronaut., vol. 171, pp. 208–214, Jun. 2020.
- 16) GitHub, C2A core, <https://github.com/ut-issl/c2a-core>.
- 17) GitHub, S2E core, <https://github.com/ut-issl/s2e-core>.