17th International Conference on Space Operations, Dubai, United Arab Emirates, 6 - 10 March 2023.
Copyright 2023 by CGI Deutschland B.V. & Co. KG. Published by the Mohammed Bin Rashid Space Centre (MBRSC) on behalf of SpaceOps, with permission and released to the MBRSC to publish in all forms.

SpaceOps-2023, ID #480

Unsupervised Hierarchical Planning for Geostationary Satellite Missions

Dr. Michael Schmeing*[a], Nicholas Symons[b]

[a] *CGI Deutschland B.V. & Co. KG, Alte Wittener Straße 56, 44803 Bochum, Germany*, michael.schmeing@cgi.com
[b] *CGI Deutschland B.V. & Co. KG, Mornewegstr. 30, 64293 Darmstadt, Germany*, nicholas.symons@cgi.com
*Corresponding Author

Abstract

A new mission planning system is introduced that provides flexible adaptation and deployment options for heterogeneous satellite missions. Modern satellite missions often consist of several user mission profiles with each one potentially having different requirements in terms of access control and data separation. External access to the planning system is required, often in combination with the desire to keep the involvement of the operations team in day-to-day planning operations as low as possible.

In this paper, a generic and flexible planning solution is presented that can be adapted to different mission profiles. In addition to the typical requirements for a satellite mission such as constraint-based de-conflicting, resource planning and planning with flight-dynamics events, three main aspects of the presented mission planning system provide a great level of flexibility and adaptability to specific mission needs: First, the system allows for access for external users. Second, the system supports lights-out operations. Third, the system allows different user mission profiles to have different levels of data separation. The mission planning system is developed by CGI Germany under the product name PLENITER Plan and will be used as the mission planning system for a geostationary communication satellite in 2023.

If a large number of external users, potentially not particularly experienced with satellite operations but rather specialists in their own payload, want to request the scheduling of on-board activities, a high level of user autonomy is desirable to reduce the workload of the operations team. To allow this, the presented mission planning system provides a flexible access-control system that allows fine-grained access control on both function and content level. In function-level access control, access to all software functions is granted or denied for configurable groups of users. In content-level access control, access is granted or denied for each entity in the mission plan: For each planning concept, i.e., task, activity, resource, constraint, etc., four levels of access control can be defined: read/write, read, anonymized, none. Access control on both function and content level is configured in the mission rules, allowing tailoring of the application to arbitrary mission needs and user groups. This also allows the re-use of the frontend application both for the operational control of the mission planning system by the operators as well as for the external users. External users therefore are granted the same level of schedule visualization as operational users, however tailored to their specific needs and access level.

Lights-out operations refers to operations without any human operator in the control center. A satellite mission could, for example, be operated only during office hours on weekdays, with no operations team in the control center during the night or on weekends. Nevertheless, during those times, user planning requests need to be received, processed and executed. To allow for quick changes in the payload configuration, the time from the ingestion of a planning request to the uplink of the respective telecommands must be as low as possible. PLENITER Plan, embedded into the PLENITER suite of ground segment software components, usually requires no more than a few seconds to process and forward a planning request to execution. This is on the one hand possible due to a powerful and efficient de-conflicting algorithm and on the other due to the usage of non-failing constraint resolving strategies: The main constraint de-conflicting algorithm of the mission planning system is optimized towards feasibility of the mission plan, i.e., trying to resolve as many scheduling constraints as possible. Starting from a feasible mission plan (i.e., a mission plan having no constraint violations), the system continues to produce feasible mission plans by applying de-conflicting strategies based on first-come-first-served basis and/or priority. In typical routine scenarios, a user planning request can be received, checked, ingested, placed, planned, checked, scheduled and executed completely automatically, without any operator interaction in under one minute. During this time, the external user is notified about every state change in the lifecycle of a task.

Copyright 2023 by CGI Deutschland B.V. & Co. KG. Published by the Mohammed Bin Rashid Space Centre (MBRSC) on behalf of SpaceOps, with permission and released to the MBRSC to publish in all forms.

Data separation can be a crucial requirement in satellite missions: Different user mission profiles might have different security classifications. In addition, if several different users share the same payload within each mission profile, data separation based on need-to-know must be employed. For this task, the mission planning system is designed as a hierarchical distributed planning system with each planning node operating only on a subset of the mission configuration and mission rules, accessible within its level of security classification and need-to-know. Each planning node generates a partial-feasible schedule, giving immediate (but preliminary) feedback to the downstream nodes or users. The executable schedule is forwarded upstream until finally, one central planning node consolidates the mission plan, performing inter-segment de-conflicting. The hierarchical distributed approach ensures data separation both on content and configuration level.

This paper describes the PLENITER Plan mission planning system and illustrates, how these three main design aspects influenced the system design. In addition, other aspects are covered, i.e., the plug-in concept of the algorithm management and the seamless mission configuration workflow, allowing for flexible re-configuration of mission rules and planning configuration during the mission to account for changing needs of the user segments.

Keywords: mission planning, user missions, distributed planning

Acronyms/Abbreviations

HMI – Human Machine Interface (usually a graphical user interface)

1. Introduction

In this paper, a new mission planning system developed by CGI Germany is presented. PLENITER Plan will be used as the main mission planning system for a geostationary satellite mission which is expected to launch in 2023. The following sections present some of the core concepts that were given special attention during design and development of the system with the aim of creating a flexible, adaptable system that is not tied to a single mission but is adaptable to different mission profiles. These core concepts are:

- Support for lights-out operations by providing a planning- and de-conflicting algorithm that is able to both operate supervised and unsupervised
- A fine-grained access control system allowing for the definition of user roles both for the operators as well as for external users
- Support for a hierarchical system deployment allowing for data separation between different user mission profiles

1.1 Architecture

The system serves as a mission planning and schedule execution system and provides four central components:

- The “Mission Configuration Editor” provides the tools to allow the editing and versioning of the objects that make up the mission configuration such as the activities and tasks to execute, as well as any constraints of the mission.
- The “Planning Server” supports the requesting and editing of task instances; detection and resolution of any constraint violations; and providing a mission timeline that indicates the current state of each requested task and its time of execution.
- The “Schedule Execution Server” triggers the execution of the activities at the specified time by interacting with the execution system
- The “Frontend Application” provides a graphical user interface to the user (internal or external) to support him in his mission goals

Figure 1 shows the main components of the system. Note that the Execution Engine is not part of the presented system but rather an arbitrary execution system, for example for the execution of procedures.

Copyright 2023 by CGI Deutschland B.V. & Co. KG. Published by the Mohammed Bin Rashid Space Centre (MBRSC) on behalf of SpaceOps, with permission and released to the MBRSC to publish in all forms.

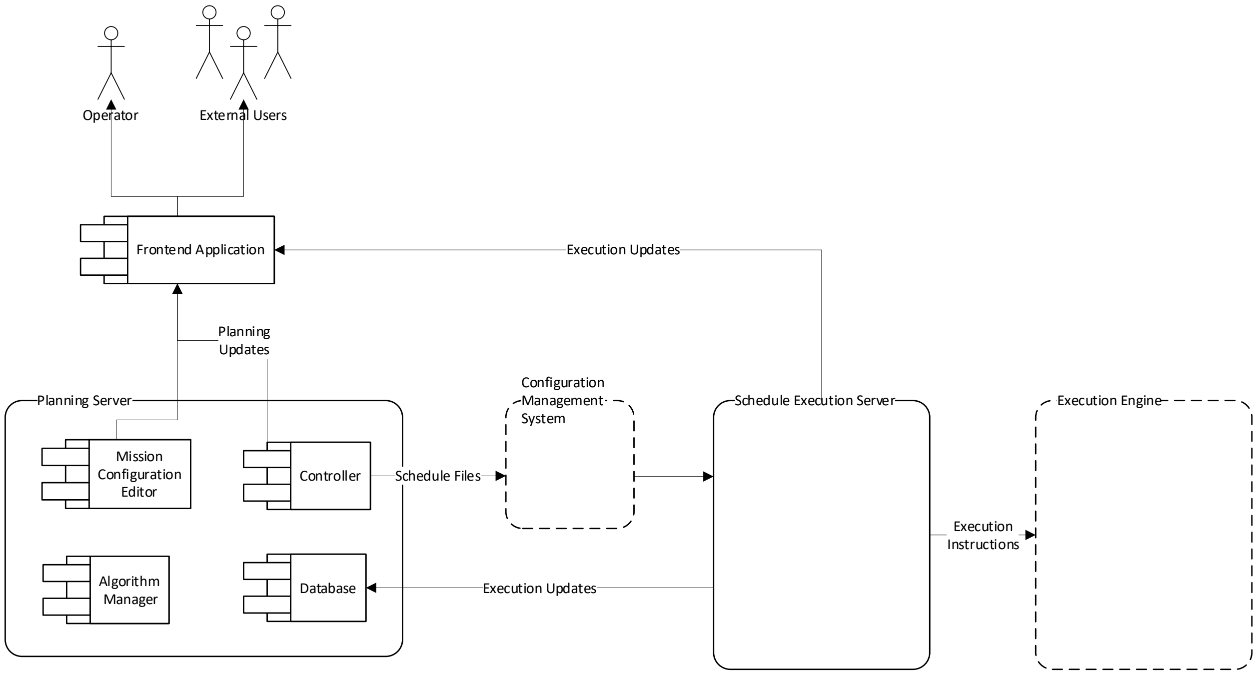


Figure 1: Basic Architecture of the System

PLENITER Plan is a server/client-based application with two backend applications, i.e., the Planning Server and the Schedule Execution Server, and a web-based frontend application. The backend applications are written in Java while the frontend application is based on Angular.

The Planning Server is the central component that provides status updates for the mission plan as well as the data from the mission configuration editor to the frontend. The Schedule Execution Server serves the frontend for online execution updates and passes them to the Planning Server for consolidation into the database.

The Planning Server consists of four components: the Mission Configuration Editor, the Controller, the Algorithm Manager and the Database. The Mission Configuration Editor is an optional component that can be left out of the deployment of the Planning Server to realize different system deployments, see also Section 1.3. The Controller coordinates the data flows between the main components. The Algorithm Manager provides facilities for the de-conflicting of the mission plan. It provides a plug-in architecture that supports arbitrary planning algorithms to be added to the system. An operator could then, for example, try out different algorithms to compare their results or employ different algorithms for separate parts of the overall planning problem. The current implementation holds one de-conflicting planning algorithm that formulates the planning problem as a constraint programming problem (see Section 3.1 for details) as well as an independent conflict checker that identifies conflicts (without resolving them). The database holds all persistent information of the planning system, i.e., the mission plan, requests, schedules as well as the mission configuration.

Apart from the HMI, the system provides file-based interfaces for ingestion of planning and event requests as well as for the exchange of schedule files via an external Configuration Management System. In addition, it can be adapted to interface with an external Execution Engine that executes the executables of which activities are comprised (see Section 1.2 for the concepts of the planning system).

1.2 Core Concepts

In the following, some of the core concepts of the system are described.

<u>Task, Activities and Events</u>

The core concepts of PLENITER Plan are *tasks* and *activities*. Activities map to atomic executables that are executed by the execution engine. Activities can be parametrized and the presented system supports a wide variety of data types by supporting the full set of SCOS data types (as defined in the Packet Utilization Standard [1]) including repeated arguments. “Atomic” means in this context that PLENITER Plan does not know about the inner workings of activities but simply tracks their execution as a whole via responses received from the Execution Engine. Tasks

Copyright 2023 by CGI Deutschland B.V. & Co. KG. Published by the Mohammed Bin Rashid Space Centre (MBRSC) on behalf of SpaceOps, with permission and released to the MBRSC to publish in all forms.

comprise of activities. Each task can hold multiple activities which can be placed within the task relative to its start or end. Events are external activities that are outside of the influence of the mission planning system such as flight-dynamics events, maintenance windows, etc. The algorithm cannot move events during de-conflicting but events can serve as boundary conditions for the planning. Tasks, activities and events are defined with the Mission Configuration Editor component.

Planning and De-conflicting

PLENITER Plan uses a planning methodology whereby planning is performed in short cycles on a small number of changes to an existing and valid plan. This allows for a very short delay between changes being made to the mission plan and their eventual execution. The short cycles also provide quick feedback to a planning engineer or external user. Similar methodologies for planning have been used in other mission planning systems: respectively referred to as “Incremental Planning” and “Continuous Planning” in [2, 3]. The planning algorithm of PLENITER Plan is a de-conflicting algorithm. Each task is requested via a planning request that allows for the definition of a request window in which the task can be executed. The algorithm moves the task within this request window when trying to create a feasible, conflict-free mission plan. In addition, a priority can be assigned to each request, providing the algorithm with more information to find a better solution in alignment with the mission goals. More about planning is described in Section 3.

Constraints

The system supports the following constraint types:

- Exclusion/inclusion constraints
- Temporal constraints
- Resource constraints

Exclusion and inclusion constraints allow the modelling of restrictions that dictate that certain entities (tasks, activities and/or events) cannot (partially) overlap, or must overlap, respectively. Temporal constraints allow for the definition of arbitrary temporal dependencies between the start and end times of entities to, for example, ensure that a certain buffer is always maintained between particular tasks. Resource constraints ensure that certain ground- or on-board resources that are modelled by PLENITER Plan always meet definable conditions, for example never exceed a given threshold. PLENITER Plan keeps track of allocated resources and projects resource usage into the future in order to identify future resource conflicts when de-conflicting new incoming planning requests.

1.3 Functions

The presented system provides the following core functions:

Automatic and Manual Planning

The system provides an automatic de-conflicting algorithm that can operate unsupervised to create a feasible mission plan that meets a set of pre-defined constraints. Here, each task request is automatically placed into the mission plan, any constraint conflicts are automatically resolved and each task is automatically scheduled for execution. Automatic planning is at the heart of lights-out operations, see Section 3 for details on the planning algorithm.

All steps that are executed by the automatic planning algorithm can also be executed by Operators, i.e., they can manually modify, de-conflict and schedule sections of the mission plan.

Schedule Execution and Execution Control

The scheduled tasks and their comprising activities will be sent at the start time of each activity to the Schedule Execution Server for execution. The system provides the ability to control the execution of the tasks and activities by suspending, stopping or resuming parts of the mission plan.

Task Requests

The system can receive task requests from two kinds of sources: internal sources (i.e., operators and external users interfacing with the HMI of the system) and external sources. The external sources interact with the system via a file-based request interface. Both sources can request tasks for execution within a particular time range with any required arguments. An example for an external source is a flight dynamics system which can request maneuver tasks via this interface as well as inject flight-dynamic events such as eclipses, sensor blindings, etc.

Copyright 2023 by CGI Deutschland B.V. & Co. KG. Published by the Mohammed Bin Rashid Space Centre (MBRSC) on behalf of SpaceOps, with permission and released to the MBRSC to publish in all forms.

Visualization of Mission Plan

The system provides the operators and external users with a visual timeline of the mission plan, indicating the current state and execution time of requested tasks, activities and events, see Figure 2. The lifecycle of each task, activity and event is modelled via a state model. Each state change is logged together with timestamp, reason and originator of the state change. This way each processing step of the planning system and schedule execution system can be followed and analyzed, see Figure 3.

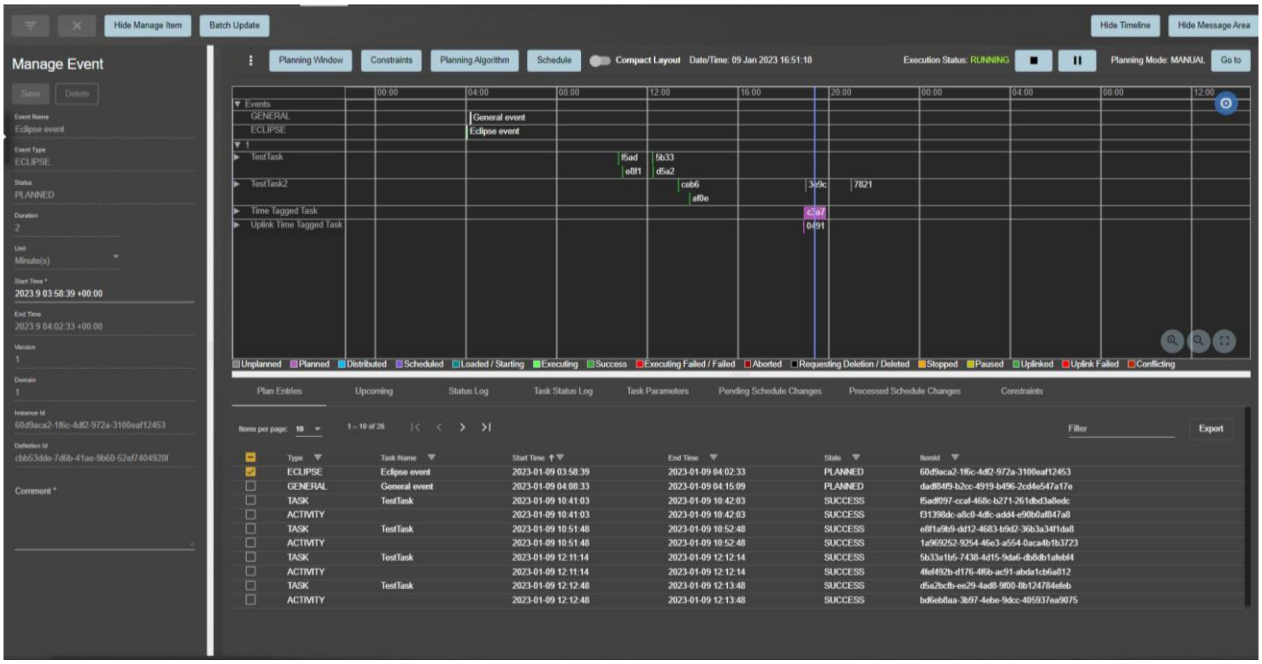


Figure 2: Main HMI of the Planning Tool with Gantt Chart and State Log

Configuration Management

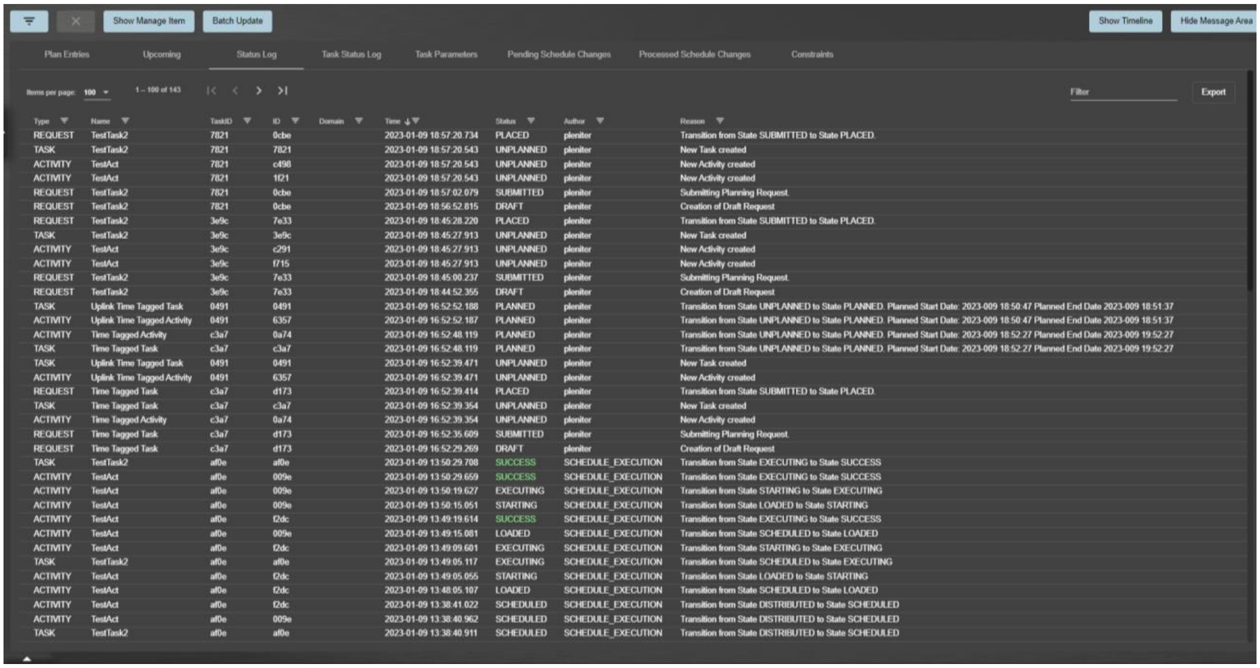
The system stores, and allows operators to edit, tasks, activities, constraints and resources of the mission. The Mission Configuration Editor provides a graphical user interface to the operators through which they can adapt the mission configuration during the mission lifetime without the need for support from the technical team. Through this mechanism, changes or additions to the mission objectives can be accounted for quickly throughout the mission. The Mission Configuration Editor also supports the versioning and baselining of the configuration set as well as the ability to export/import to/from an external configuration archiving system.

Different deployments of the Mission Configuration Editor are possible in order to account for different chain deployments. In general, the Mission Configuration Editor can be deployed directly on a planning server to allow for in-place editing of the mission configuration. Via access control, the Mission Configuration Editor can be made read-only (for certain user groups) and therefore serve as a task library, showing the available tasks for the specific user groups. Typically, however, a mission has a dedicated validation environment for task development and validation with a Mission Configuration Editor deployed to the planning server of the validation environment. The operational chain that controls the satellite does not have the Mission Configuration Editor deployed. The mission configuration of the planning system is then deployed via a configuration archiving system.

Data Separation and Protection

The system provides mechanisms to limit the ability of particular operators to access certain data and operations. The mechanisms include the separation of data into different planning nodes and also a permissions system to determine the operations and data visibility to which each operator is entitled. See Section 4 for details.

17th International Conference on Space Operations, Dubai, United Arab Emirates, 6 - 10 March 2023.
Copyright 2023 by CGI Deutschland B.V. & Co. KG. Published by the Mohammed Bin Rashid Space Centre (MBRSC) on behalf of SpaceOps, with permission and released to the MBRSC to publish in all forms.



Type	Name	TaskID	ID	Domain	Time	Status	Author	Reason
REQUEST	TestTask2	7821	0cbe		2023-01-09 18:57:20.734	PLACED	pleniter	Transition from State SUBMITTED to State PLACED.
TASK	TestTask2	7821	7821		2023-01-09 18:57:20.543	UNPLANNED	pleniter	New Task created
ACTIVITY	TestAct	7821	c498		2023-01-09 18:57:20.543	UNPLANNED	pleniter	New Activity created
ACTIVITY	TestAct	7821	1f21		2023-01-09 18:57:20.543	UNPLANNED	pleniter	New Activity created
REQUEST	TestTask2	7821	0cbe		2023-01-09 18:57:02.079	SUBMITTED	pleniter	Submitting Planning Request.
REQUEST	TestTask2	7821	0cbe		2023-01-09 18:56:52.815	DRAFT	pleniter	Creation of Draft Request
REQUEST	TestTask2	3e9c	7e33		2023-01-09 18:45:28.220	PLACED	pleniter	Transition from State SUBMITTED to State PLACED.
TASK	TestTask2	3e9c	3e9c		2023-01-09 18:45:27.913	UNPLANNED	pleniter	New Task created
ACTIVITY	TestAct	3e9c	c291		2023-01-09 18:45:27.913	UNPLANNED	pleniter	New Activity created
ACTIVITY	TestAct	3e9c	f715		2023-01-09 18:45:27.913	UNPLANNED	pleniter	New Activity created
REQUEST	TestTask2	3e9c	7e33		2023-01-09 18:45:00.237	SUBMITTED	pleniter	Submitting Planning Request.
REQUEST	TestTask2	3e9c	7e33		2023-01-09 18:44:52.355	DRAFT	pleniter	Creation of Draft Request
TASK	Uplink Time Tagged Task	0491	0491		2023-01-09 16:52:52.188	PLANNED	pleniter	Transition from State UNPLANNED to State PLANNED. Planned Start Date: 2023-009 18:50:47 Planned End Date 2023-009 18:51:37
ACTIVITY	Uplink Time Tagged Activity	0491	6357		2023-01-09 16:52:52.187	PLANNED	pleniter	Transition from State UNPLANNED to State PLANNED. Planned Start Date: 2023-009 18:50:47 Planned End Date 2023-009 18:51:37
ACTIVITY	Time Tagged Activity	c3a7	0a74		2023-01-09 16:52:48.119	PLANNED	pleniter	Transition from State UNPLANNED to State PLANNED. Planned Start Date: 2023-009 18:52:27 Planned End Date 2023-009 19:52:27
TASK	Time Tagged Task	c3a7	c3a7		2023-01-09 16:52:48.119	PLANNED	pleniter	Transition from State UNPLANNED to State PLANNED. Planned Start Date: 2023-009 18:52:27 Planned End Date 2023-009 19:52:27
TASK	Uplink Time Tagged Task	0491	0491		2023-01-09 16:52:39.471	UNPLANNED	pleniter	New Task created
ACTIVITY	Uplink Time Tagged Activity	0491	6357		2023-01-09 16:52:39.471	UNPLANNED	pleniter	New Activity created
REQUEST	Time Tagged Task	c3a7	d173		2023-01-09 16:52:39.414	PLACED	pleniter	Transition from State SUBMITTED to State PLACED.
TASK	Time Tagged Task	c3a7	c3a7		2023-01-09 16:52:39.354	UNPLANNED	pleniter	New Task created
ACTIVITY	Time Tagged Activity	c3a7	0a74		2023-01-09 16:52:39.354	UNPLANNED	pleniter	New Activity created
REQUEST	Time Tagged Task	c3a7	d173		2023-01-09 16:52:35.609	SUBMITTED	pleniter	Submitting Planning Request.
REQUEST	Time Tagged Task	c3a7	d173		2023-01-09 16:52:29.269	DRAFT	pleniter	Creation of Draft Request
TASK	TestTask2	af0e	af0e		2023-01-09 13:50:29.708	SUCCESS	SCHEDULE_EXECUTION	Transition from State EXECUTING to State SUCCESS
ACTIVITY	TestAct	af0e	009e		2023-01-09 13:50:29.659	SUCCESS	SCHEDULE_EXECUTION	Transition from State EXECUTING to State SUCCESS
ACTIVITY	TestAct	af0e	009e		2023-01-09 13:50:19.627	EXECUTING	SCHEDULE_EXECUTION	Transition from State STARTING to State EXECUTING
ACTIVITY	TestAct	af0e	009e		2023-01-09 13:50:15.051	STARTING	SCHEDULE_EXECUTION	Transition from State LOADED to State STARTING
ACTIVITY	TestAct	af0e	f2dc		2023-01-09 13:49:19.614	SUCCESS	SCHEDULE_EXECUTION	Transition from State EXECUTING to State SUCCESS
ACTIVITY	TestAct	af0e	009e		2023-01-09 13:49:15.081	LOADED	SCHEDULE_EXECUTION	Transition from State SCHEDULED to State LOADED
ACTIVITY	TestAct	af0e	f2dc		2023-01-09 13:49:09.601	EXECUTING	SCHEDULE_EXECUTION	Transition from State STARTING to State EXECUTING
TASK	TestTask2	af0e	af0e		2023-01-09 13:49:05.117	EXECUTING	SCHEDULE_EXECUTION	Transition from State SCHEDULED to State EXECUTING
ACTIVITY	TestAct	af0e	f2dc		2023-01-09 13:49:05.055	STARTING	SCHEDULE_EXECUTION	Transition from State LOADED to State STARTING
ACTIVITY	TestAct	af0e	f2dc		2023-01-09 13:48:05.107	LOADED	SCHEDULE_EXECUTION	Transition from State SCHEDULED to State LOADED
ACTIVITY	TestAct	af0e	f2dc		2023-01-09 13:38:41.022	SCHEDULED	SCHEDULE_EXECUTION	Transition from State DISTRIBUTED to State SCHEDULED
ACTIVITY	TestAct	af0e	009e		2023-01-09 13:38:40.962	SCHEDULED	SCHEDULE_EXECUTION	Transition from State DISTRIBUTED to State SCHEDULED
TASK	TestTask2	af0e	af0e		2023-01-09 13:38:40.911	SCHEDULED	SCHEDULE_EXECUTION	Transition from State DISTRIBUTED to State SCHEDULED

Figure 3: State Log

In addition to these core concepts, the system supports the following functionality:

- Time-tagged commanding
  - The system allows for the planning and tracking of activities that are not scheduled for immediate execution but for later execution in the on-board schedule of a spacecraft.
- Resource modelling
  - The system keeps track of definable resources. The resources to be modelled can be defined via the Mission Configuration Editor and can therefore be easily adapted by the operators during the mission. Resources are propagated and calibrated via an interface to a telemetry provisioning system. The resource consumption of an activity can either be pre-defined as fixed consumption or delegated to an external resource model together with activity arguments to compute the resource consumption of each activity instance individually. The system performs bookkeeping on future resource allocations and can enforce operator-definable constraints (via the Mission Configuration Editor) on the resources
- Extended Data Type Support
  - The system supports a wide range of data types, i.e., all data types according to the SCOS MIB ICD, including repeaters. The arguments are checked for min/max values and general sanity checks and passed to the underlying executable of each activity

In the following, the focus is put on three properties of the presented system: Access control to enable external access to the system, unsupervised planning to enable lights-out operations and hierarchical planning to allow for advanced data separation.

Copyright 2023 by CGI Deutschland B.V. & Co. KG. Published by the Mohammed Bin Rashid Space Centre (MBRSC) on behalf of SpaceOps, with permission and released to the MBRSC to publish in all forms.

2. Access Control for External Access

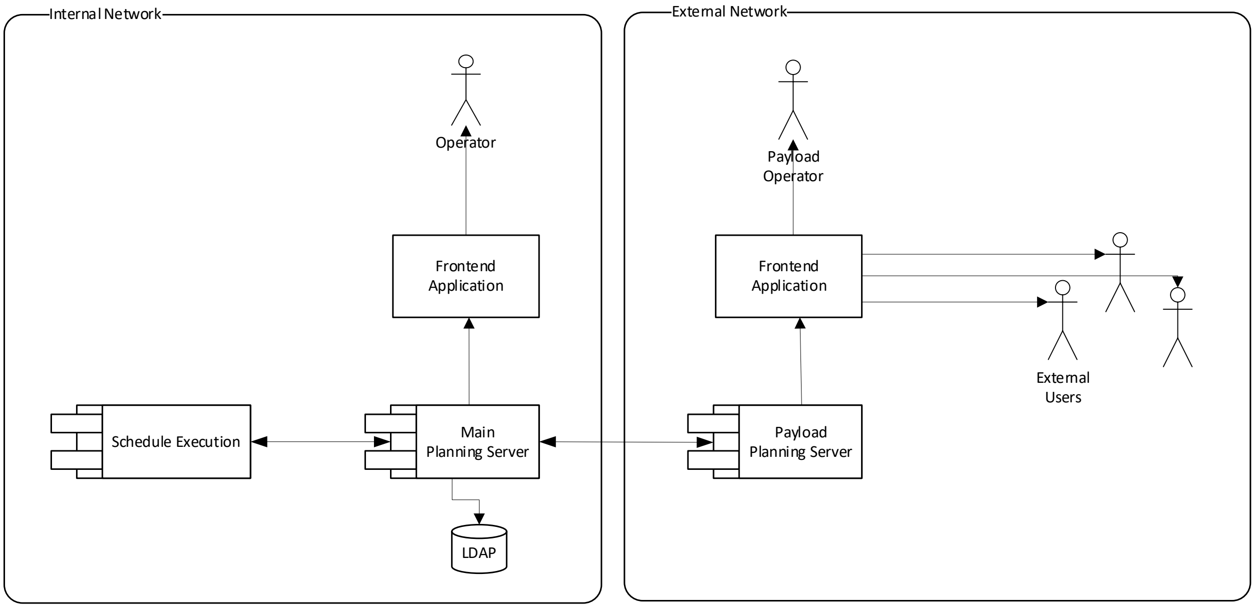


Figure 4: Deployment with external access

Satellite missions with one or multiple user mission profiles need to account for the fact that the users of the payload might not be familiar with the intricacies of satellite operations but rather just have a limited view on their own payload and the mission goals they want to achieve. Especially with a larger number of different external users, coordination and planning of the mission profiles might lead to a significant workload for the operations team. A software system that supports operations by providing a flexible, tailored and unsupervised planning and execution approach can reduce this workload.

The core concept in PLENITER Plan for external access of external users is the access control system. Access control allows external users to use the same software components for issuing planning requests and tracking their execution as the operators thus reducing the number of required components in the system and allowing for better support by the operations team. Access is limited to the functionality that is needed for the operation of the payload components.

2.1 Access Control Types

PLENITER Plan has two types of access control: function-level access control and content-level access control.

Function-level access control refers to the ability to define permissions for system functions such as conflict resolution, editing/deleting of a task, issuing a planning request and so on. Function-level access control is controlled in the backend by the planning server. The backend also controls which frontend components are visible, disabled or hidden, thus allowing to customize an HMI tailored to each user's needs and access level. This way, for example, a very limited HMI for an external user can be configured that only allows issuing of planning requests, viewing of the Gantt chart and receiving updates regarding the execution states.

Content-level access control refers to the ability to define permissions for content of the mission plan. Each entity (tasks, activities, events but also constraints and resources) is subject to content-level access control. The following levels of access are possible to configure:

- FULL: The user has read and write access to this object and its associated data
- READONLY: The user can view the object and its associated data but cannot change it or create new instances of it.
- ANONYMIZE: The user can view some data associated with the definition but cannot change it.
- NONE: The user has no access to the object or its associated data.

The access control levels can be configured via the mission configuration editor during the mission and can be freely adapted to the (changing) needs during the mission. With content-level access control, it can be ensured that

Copyright 2023 by CGI Deutschland B.V. & Co. KG. Published by the Mohammed Bin Rashid Space Centre (MBRSC) on behalf of SpaceOps, with permission and released to the MBRSC to publish in all forms.

each external user only has access to his own tasks and activities. Depending on the configured access levels of other entities, he can see and edit those entities (FULL), only see them (READONLY) or will not see their existence at all (NONE). A special case is ANONYMIZE. Here, the external user can see only the existence of a task/activity (i.e., start and end time) but not their name, arguments or owner, see Figure 5. This way, it is signaled that there is "something going on" during a given timespan, but not what exactly. This can be useful for missions where multiple parallel running tasks might result in the rejection of another task requested during that time period. An external user can then improve his chances that his planning request gets accepted by choosing a less crowded time.

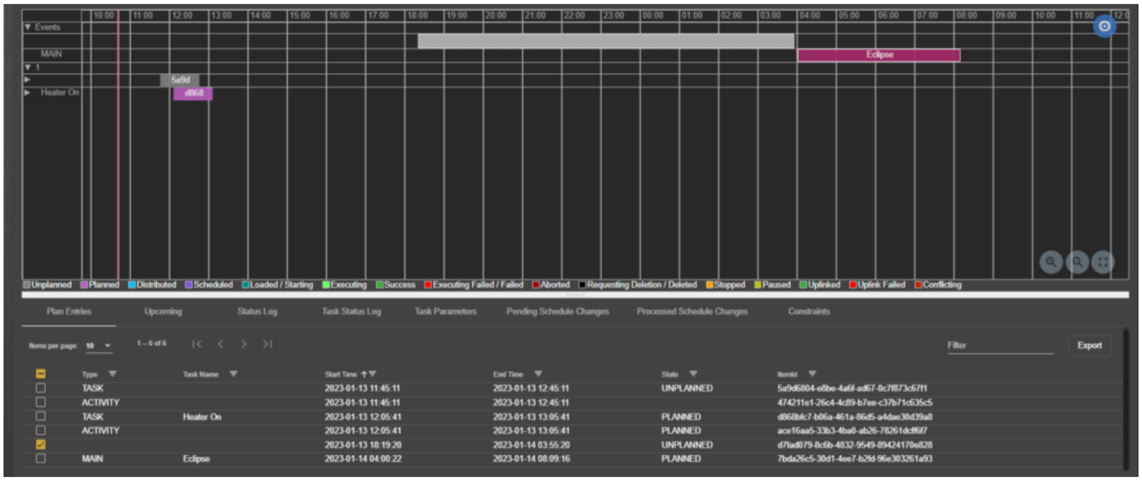


Figure 5: Anonymized View of the Mission Plan: For some tasks/activities only their existence is shown

2.2 Permissions

Each object has an associated set of permissions. The permissions consist of the following information:

- Owner: The name of the LDAP group/role that is considered the owner of the object. The owner has complete permissions to the object.
- Group: The name of the LDAP group/role that has specific and configurable access to the object. The level of access is defined by the following property.
- Group Permission: The level of access that any user within the specific Group has for the object. If this access is more restrictive than the Public Permission then it is ignored.
- Public Permission: The level of access for any user that does not belong to either the Owner or Group categories.

The Access Control will always provide the highest level of access that is possible given the membership of the user to the different groups/roles. For example, if a user belongs to both the Group and the Owner Category then the user will have full access to the definition, regardless of how restrictive the Group Permissions are. Permissions are stored in an LDAP database and new users can be added by creating the respective LDAP user. Figure 4 shows an example deployment for a planning system with external user access.

2.3 Example User Definitions

Typical user definitions include a *Planner* role with full planning permissions for all types of tasks and activities. If needed, the Planner can be limited to planning only, restricting the actual distribution of a schedule to the Schedule Execution system to a *Supervisor* role. *External Users* are limited both on function-level and on content-level to only be able to interact with their own tasks. This way, also advanced concepts like pre-defined planning priorities (see Section 3) for different user groups can be realized: Each planning request can be given a default priority via the Mission Configuration Editor. The ability to change the priority during issuing a planning request can be revoked from an external user so that he falls back to the default priority.

Copyright 2023 by CGI Deutschland B.V. & Co. KG. Published by the Mohammed Bin Rashid Space Centre (MBRSC) on behalf of SpaceOps, with permission and released to the MBRSC to publish in all forms.

3. Unsupervised Planning for Lights-out Operations

The planning system supports two operational modes: MANUAL and AUTOMATIC. In manual mode, each step in the planning process is executed by the operator, i.e., adding requests to the mission plan, checking constraints and de-conflicting them as well as distributing schedules to the schedule execution component. In automatic mode, all these steps are automatically executed by the system in configurable recurring time intervals. This way, each planning request is regularly added to the mission plan, de-conflicted and distributed. A configurable dead-time defines the latest point in time before “now” when the system accepts a planning request to allow for the algorithm to run and for the forwarding to schedule execution. In the current system, the dead-time can be set to less than 1 minute meaning that planning requests (both from internal and external users) can be processed, checked and forwarded to execution almost immediately.

3.1 Planning Algorithm

The planning algorithm solves planning problems by modelling the planning schedule as a constraints programming problem and invokes a third party constraints programming library to solve it. Each Task, Event, Requested Task, Activity and Requested Activity are represented by the following three variables. Only some variables are 'solvable' for the algorithm - meaning that the algorithm will attempt to find a solution for it:

- Start Time: This variable is a solvable one whose range of values is the requested start window
- End Time: This variable is at a fixed offset from the start time and therefore is not solvable but is used to define certain constraints
- Is-Deleted Flag: This variable is a solvable one that represents whether this item is active or deleted.

Some items within a schedule are presented as *fixed entries* to the algorithm. These items are taken into consideration by the algorithm but cannot be moved or deleted. They are modelled by making all three of the variables unsolvable by the algorithm. Examples of Fixed Entries include Events and Tasks in the DISTRIBUTED state, i.e. tasks that are currently being forwarded from the planning server to the schedule execution component and are thus not under control of either component.

Resources are modelled by a set of time-value pairs indicating the initial state of the resource as well as any change during the timeframe of the schedule being solved.

Constraints (i.e., exclusion/inclusion constraints, temporal constraints and resource constraints) are represented by basic logic structures which, when evaluated to true, indicate that the constraint is satisfied. They are built on one or more of the three variables that represent two items or resource values in the schedule.

3.2 Unsupervised Conflict Resolution Strategies

When entering a planning request, the requester is asked to select a preferred start time but also an earliest and latest start time. Earliest and latest start time form the request window, i.e., the “wiggle room” that the planning algorithm will make use of to meet the defined constraints. However, there will still be situations where the current set of constraints cannot be met by the given tasks and task requests. In this case a conflict resolution strategy is needed. Especially for unsupervised automated planning, a non-failing conflict resolution strategy is needed that will always produce a valid mission plan. Only then can interruption-free operations be achieved outside of office hours.

The system supports two factors that aid the conflict resolution: priority and submission date. Priority is a number between 0 and 100 and can be given to each task when requesting it. The submission date is the date when the planning request was received by the system. The system can use these two factors or a combination of both to determine to which task to give precedence in case of conflicts. If during constraint resolution an unresolvable conflict between two tasks is found, then the system can, for example, give precedence to the task with higher priority. If the yielding task is in request state, then the request is rejected. If it is already part of the mission plan, the task is deleted. To avoid deletion of tasks, the submission time can also be used to resolve unresolvable conflicts. Here, the task that was submitted earlier takes precedence, regardless of the priority (“first-come-first-serve”). Note that using the submission time prevents tasks from being deleted as existing tasks will always have precedence over incoming requests and thus in most circumstances will never be deleted (unforeseen event additions into the mission plan with constraints on them might cause old tasks to be deleted). Finally, a combination of both factors can also be used: the first factor is the priority and only when two conflicting tasks have same priority will the submission time be taken into account.

The utility of these resolution strategies is naturally dependent on sensible and feasible constraint definitions. Contradictory constraints or constraints based on unpredictable events can hinder or prevent effective conflict

Copyright 2023 by CGI Deutschland B.V. & Co. KG. Published by the Mohammed Bin Rashid Space Centre (MBRSC) on behalf of SpaceOps, with permission and released to the MBRSC to publish in all forms.

resolution. Aside from this caveat, the usage of any of these conflict resolution strategies ensures that the planning algorithm will always find a feasible solution to the de-conflicting problem. This way, operations during lights-out periods can also continue without the need of operator interaction. The conflict resolution strategies are communicated to all external users and the system gives proper indication on what the reason for the rejection (or deletion) of a task was, so that users can at all times understand the underlying principles of the planning decisions.

3.3 Performance

Table 1 shows example runtimes for the planning algorithm for different combinations of tasks and constraints. Note the increased runtime when task deletion is expected since task deletion is considered only as a last-resort solution when any other options has failed.

Table 1: Performance of the Planning Algorithm

		Run times without task deletion (sec)			
		# of Constraints			
		500	1000	1500	2000
# of Tasks	2000	1.6	1.8	2	2
	4000	1.8	2	2.1	2.3
	6000	2.4	2.4	2.4	2.4
	8000	3.4	3.6	4	3.4
	10000	5.5	4.6	4.4	4.5

		Run times with task deletion (sec)			
		# of Constraints			
		500	1000	1500	2000
# of Tasks	2000	0.5	0.4	0.5	0.7
	4000	1.7	1.9	1.7	1.8
	6000	3.9	6.4	6.7	4.9
	8000	12.5	14.9	11.4	9.2
	10000	23.1	20.9	17.5	20

4. Hierarchical Planning for Advanced Data Separation

The access control system described in Section 2 allows for data separation on user interface level: the planning server still has full knowledge of the underlying mission plan and the mission configuration but only provides a subset of that information to each user, based on the user role and the defined set of permission. In some scenarios, though, an advanced data separation might be necessary where the planning server does not have the full visibility of the complete mission plan and its underlying mission configuration. In these cases, a hierarchical planning approach is needed to allow for the generation and merging of feasible sub-schedules to a consolidated mission plan.

A typical example for such a deployment could be a mission with multiple user mission profiles that have different security classifications. Not all information of the high-security segments of the system are allowed to enter the low-security segments. This applies both to the mission plan and the execution status of the different entities as well as to the mission configurations of the different mission profiles. In this case, it is desirable that each mission profile can perform a preliminary planning session on their own mission profile schedule “to its best knowledge”, i.e., with all constraints that are applicable to the particular user mission profile. However, a central planning component will have the last word on the mission plan, possibly enforcing additional constraints unknown to the mission profiles. This approach is called “hierarchical planning”.

4.1 Planning Nodes

The presented planning system supports hierarchical planning in the following sense: instead of a single planning server, a set of multiple planning servers (called “planning nodes”) is deployed. The planning nodes are configured in a tree-like topology, i.e., each having exactly one parent node and zero to multiple child nodes. The single node without a parent node, the root node, is the only node connected to an execution component, in this case the Schedule Execution Component. Figure 6 shows an example deployment.

Copyright 2023 by CGI Deutschland B.V. & Co. KG. Published by the Mohammed Bin Rashid Space Centre (MBRSC) on behalf of SpaceOps, with permission and released to the MBRSC to publish in all forms.

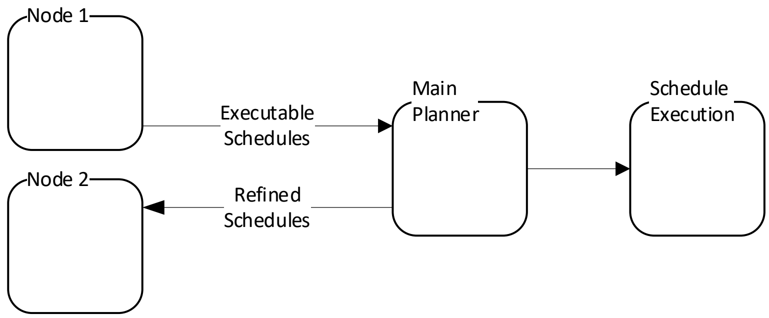


Figure 6: Chaining Deployment

Planning nodes pass executable schedules upstream (i.e., towards their parents) and pass refined schedules downstream (i.e., towards their children). Executable schedules contain consolidated sub-schedules, i.e., schedules that are conflict-free according to the planning configuration of the current node while refined schedules contain the delta-changes that an upstream node had to make to each sub-schedule to account for the delta-constraints that it exclusively has.

The use of distributed and hierarchical planning nodes has been used in other planning systems, such as the one described in [4], but in order to achieve different ends. The architecture outlined in [4] has the main goal of accommodating the large geographical distribution of payload users and the diverse uses of the payload as opposed to the separation of data.

4.2 Hierarchical Conflict Resolution

The hierarchical planning configuration must still be able to support automatic de-conflicting for lights-out operations. For this, the unsupervised conflict resolution strategies, described in Section 3.2, are also employed in the hierarchical planning deployment. In addition, however, another implicit conflict resolution strategy is employed: The (single) parent node is always right, i.e., it takes precedence in case of conflicts. This mirrors a system topology where higher-level planning nodes relate to higher-level system functions, i.e., the main planning node is usually dealing with satellite platform planning while its child nodes deal with mission profile planning. Satellite platform will ultimately always take precedence over the user mission profiles when it comes to satellite safety. It can give precedence to the user mission profiles so that some non-urgent platform activity is moved to a point in time where it does not interfere with the user mission profile, but this decision is still taken by the platform: never will a user mission profile take precedence when the satellite platform actually wanted to have precedence. This typical system topology is reflected by the “parent is always right” conflict resolution strategy.

For this to result in conflict-free sub-schedules in the child nodes, the mission configuration (which includes the planning rules and the defined constraints) that is deployed to each child node must always be a subset of the mission configuration of the parent node. This way, a feasible schedule generated upstream at the parent node might still be different from the downstream-generated schedule of the child node but since it covers the same constraints deployed at the child node (and potentially more) it is still feasible, i.e., conflict-free at the child node. The child node, upon receiving the refined schedule from the parent node, might still need to adapt its local mission plan but this will only result in a change from one feasible schedule to another feasible schedule.

If the parent node finds a non-resolvable conflict at his level, a conflict resolution decision (usually the deletion of a task) will be reported downstream to the child node. The child node might not comprehend the decision to delete a certain task since this is based on a constraint that is not available on child level. However, it will nevertheless delete the task from its mission plan which will still result in a feasible mission plan on the child level.

All in all, the usage of the “parent knows best” conflict resolution strategy ensures that no loops of back-and-forth sending of executable schedules and refined schedules occurs – the hierarchical conflict resolution converges after one iteration with the parent.

<u>Global State Log</u>

In service of the “parent knows best” conflict resolution strategy, a global state log is maintained across the entire deployment. The global state log holds every change that was made to the mission plan and a provisional global state log is propagated both upstream and downstream alongside the schedule by each planning node. This mechanism allows any conflicts and inconsistencies between an incoming schedule and a local one to be directly detected and resolved. To enforce the “parent knows best” strategy, an inconsistency detected in an incoming upstream schedule

Copyright 2023 by CGI Deutschland B.V. & Co. KG. Published by the Mohammed Bin Rashid Space Centre (MBRSC) on behalf of SpaceOps, with permission and released to the MBRSC to publish in all forms.

results in the inconsistent local changes being rolled back and, conversely, inconsistencies found in an incoming downstream schedule results in the rejection of those incoming changes. Through this process, a globally recognized state log is established.

4.3 Performance

The performance of the hierarchical planning configuration scales with the performance for the single planning node described in Section 3.3 and linearly depends on the number of planning nodes deployed along the longest path in the deployment tree. This is ensured due to the fact the hierarchical conflict resolution described in the previous section ensures that on each level of the planning hierarchy, only one planning session needs to be executed. Refined schedules that downstream nodes receive do not trigger another planning session but just require the realization of the changes done by the parent node.

5. Conclusions and Outlook

In this paper, a flexible and adaptable planning solution has been presented that can be adapted to various mission profiles. A fine-grained configurable access control system allows the reuse of the core planning components for both operator and external user access, which simplifies system development for missions with multiple external users. Lights-out operations is possible due to an automatic de-conflicting algorithm that always finds a feasible mission plan based on a configurable set of conflict resolution strategies. Finally, hierarchical planning allows an even further separation of data from both the mission plan and configuration.

The presented planning system will be used as the main planning system for a geostationary mission in 2023 comprising of multiple user segments with multiple external users. As of now, the system is focused on single-satellite missions, so the future work concentrates on adding capabilities to deal with constellation planning. The underlying planning algorithm is already in principle capable of dealing with planning problem of non-geostationary multi-satellite missions, these capabilities are currently refined and being made available through suitable workflows and processes in the HMI.

Acknowledgements

Special thanks to the development team, i.e., Pavlos Milaszewicz, Peter Danko, Martin Janoušek, Andrea Suchánková, Hana Šimková, Giovanni Amoriello, Max Reppin, Timo Schroth and Jan Nörtemann.

References

[1] "Telemetry and telecommand packet utilization," ECSS-E-ST-70-41C, ESA-ESTEC, Noordwijk, The Netherlands, 2016, 15 April

[2] T. M. Wörle, C. Lenzen, T. Göttfert, A. Spörl, B. Grishechkin, F. Mrowka, M. Wickler, "The Incremental Planning System – GSOC's Next Generation Mission Planning Framework," SpaceOps2014 13th International Conference on Space Operations, Pasadena, California, USA, 2014, 5 – 9 May.

[3] S. Chien, G. Rabideau, R. Knight, R. Sherwood, B. Engelhardt, D. Mutz, T. Estlin, B. Smith, F. Fisher, T. Barrett, G. Stebbins, D. Tran, Aspen-automated planning and scheduling for space mission operations. Space Ops Vol. 82. (2000)

[4] J Hagopian, T Maxwell, T Reed, "A distributed planning concept for space station payload operations" Third International Symposium on Space Mission Operations and Ground Data Systems, Goddard Space Flight Center, 1994, 15-18 November