Evolution of a Flexible Mission Planning and Scheduling System for Complex Missions: *flexplan*

Alicia T. Kavelaars[1], Assaf Barnoy[1], Francisco Colmenero[2], María Pereda[2], Juan Tejo[2], Gonzalo Garcia[1], Theresa W. Beech[1]

[1] GMV Space Systems Inc., 1375 Piccard Dr., Suite 250, Rockville, MD 20850, USA, +1 (301) 216 3840
akavelaars@gmvspacesystems.com

[2] GMV S.A., Calle Isaac Newton, 11, PTM-Tres Cantos, 28760 Madrid, SPAIN, +34 91 807 2100

Abstract

The space industry has evolved in the past decades with the result that the main satellite operators are in charge of fleets of spacecraft manufactured by different companies with different characteristics. Additionally, many new scientific missions have come up with the quest to reach Mars and the Moon as well as a better knowledge of the Earth, with ever improving instruments and increased data recordings. The operation of multiple platform fleets or highly specialized missions have one thing in common: they require a tool that can plan and schedule while adapting to the ever changing requirements and constraints driven by the mission. ***flexplan*** is a COTS Mission Planning and Scheduling (MPS) product that has matured in the past years while integrated in such missions. Its primary goal of offering a flexible solution that can adapt with minor operational impact has been achieved and superseded by an increased focus on performance when multiple platforms and complex operations are present. This paper describes the latest upgrades as well as a sample application of *flexplan*.

I. Introduction

Mission planning and scheduling is a very dynamic process in spacecraft operations. It is the process by which inputs from the spacecraft operator and user community requests are gathered, managed and scheduled. This process is complicated by the changing priorities, availability of resources and operational constraints that are part of daily operations. Some of the elements that must be managed by the Mission Planning System include:

- **Resource scheduling:** The operations and maintenance of on-board and ground resources must be managed to avoid delays or losses due to the oversubscription of mission resources. The availability of personnel must also be considered to ensure proper support of mission critical events.

- **Instruments operations:** The on-board instruments maintenance and science operations may vary according to target of opportunity or the operational status of the instruments. Requests from the various instrument teams must be prioritized, scheduled and de-conflicted if necessary.

- **Satellite operations:** Satellite operation activities must be managed according to the different phases of the mission and to the availability of communication links to the spacecraft. It must also be possible to incorporate changes in constraints, permanently or temporarily, due to changes to the scientific mission, or changes in the operational status of spacecraft systems.

- **Ground operations:** Ground operations activities must be managed as well as adapting to any changes in operations that may occur in the course of the mission (e.g. new ground stations added to the ground network, automation may be implemented).

The scope of the Mission Planning depends highly upon the mission. Inputs and outputs can vary quite widely. Some Mission Planning systems accept user requests, while others do not. Some Mission Planning systems generate only a high level plan, while others generate a low level command procedure schedule, and yet others will even monitor and control the execution of the schedule. However, for all Mission Planning systems there is a common point: they all follow the Mission Reference Operations Plan (ROP), which describes the different scenarios that the mission will encounter during its execution. (Please see the next section for a more detailed description of the ROP).

The ideal situation when implementing any software tool is that the inputs and outputs are known beforehand completely, and are frozen. The programmers can then implement the SW in a straightforward way, without necessarily having any specific detailed knowledge of the mission.

However, this is not a realistic situation for most Mission Planning systems. The main input for the Mission Planning system is the ROP, and the ROP can evolve

significantly over the development of the mission, as well as during the mission lifetime. In addition, the design and implementation of the Mission Planning may, and usually does, start before the ROP has been fully written.

This situation leads to a problem. The challenge is to develop a cost-efficient Mission Planning system whose inputs are not completely known during development, while not running into time-consuming and costly changes in the SW for every change.

In order to address this problem, GMV has developed a Generic Mission Planning kernel called *flexplan*.

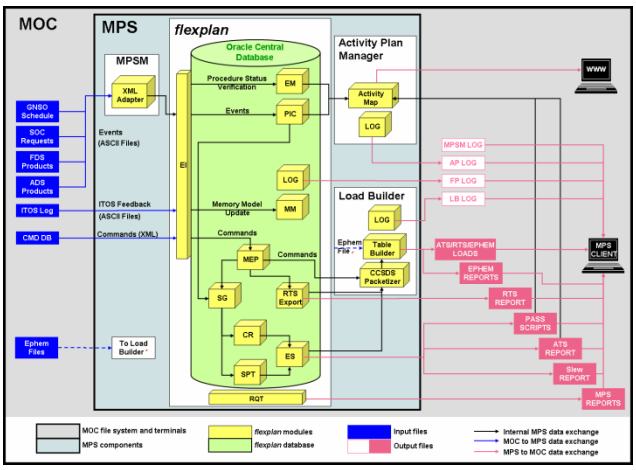

Figure 1. *flexplan* Architecture

II. *flexplan* Overview

flexplan is a highly configurable tool which can be efficiently configured for different missions. It covers the end-to-end loop of Mission Planning and Scheduling and allows users to adapt the system to their requirements quickly and easily.

At the core of *flexplan*'s scheduling process is a Soft Algorithm Generation engine that requires no recompiling of the tool whenever flight rules change. This engine is largely responsible for the ease of adaptability of *flexplan* to the different mission phases, requirements, and styles of planning and scheduling operations. This soft algorithm generation engine can be used without needing to have any programming background. The only operational constraint is to be able to identify the rules that will plan the daily operations of the spacecraft and the ground activities during the different phases of the mission.

flexplan's modular architecture allows its components to interact with each other via a database. This allows different components to be run at different times or concurrently by different operators. This architecture also allows *flexplan* to be easily extended with additional modules to support specific mission requirements or needs.

Figure 1 shows a *flexplan*'s architecture. The following sections briefly describe *flexplan* Mission Planning System components. They are divided into two categories:

- **Core components**: Modules responsible for the generation of mission conflict free schedules.
- **Optional components**: Modules supporting additional features for the status awareness of planned activities.

A. Core Components

All interfaces between *flexplan* and the external world are managed by this subsystem. It has a "pluggable" architecture which uses an XML interface schema (which is publicly available) to adapt easily to different types of missions.

1. Mission Environment Preparation (MEP). This offline tool is used to configure the mission. Using the soft algorithms embedded in the MEP, the user is able to define the flight rules and mission rules for a specific mission, along with the events, resources (including availability profiles), tasks and operations of the mission. Using this information, the MEP generates the Master Schedules which define specific scenarios within the mission and break the mission up into as many scenarios as the user specifies. The MEP is primarily configured the beginning of a mission and thereafter is run only whenever the mission or flight rules change. All the elements involved in the mission planning process can be defined and changed by the operations team during the operational lifetime of the mission.

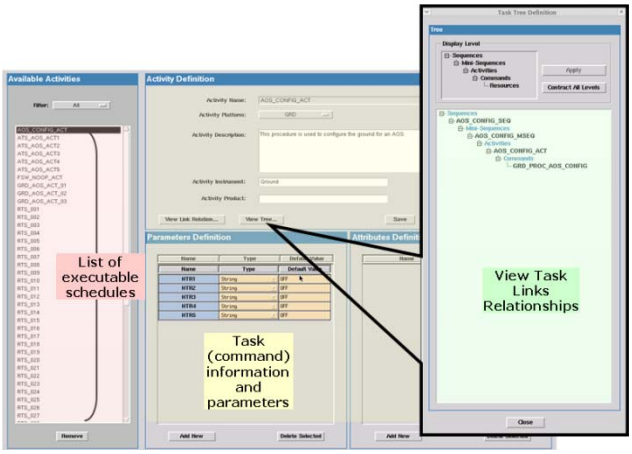

Figure 2. *flexplan* Mission Environment Preparation GUI (MEP)

2. Planning Input Customization (PIC). During mission operations, this *flexplan* subsystem gathers all external inputs which may trigger events for the mission planning system. These inputs include Flight Dynamics events (i.e. eclipses, ground station acquisition and loss, maneuvers), user requests from the scientific community, and any user defined events as specified.

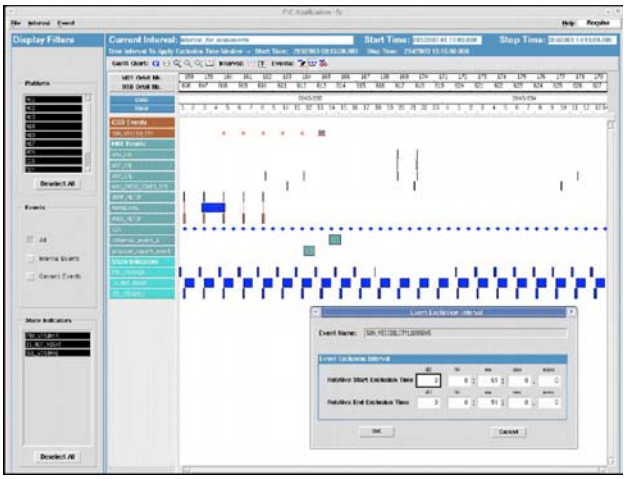

Figure 3. *flexplan* Planned Input Customization GUI (PIC)

3. Schedule Generator (SG). Using the inputs from the PIC and a Master Schedule from the MEP, this *flexplan* subsystem generates a mission schedule. This schedule contains the operations that are linked to the events defined by the user via the soft algorithms. The schedule generated by this subsystem may or may not be conflict free.

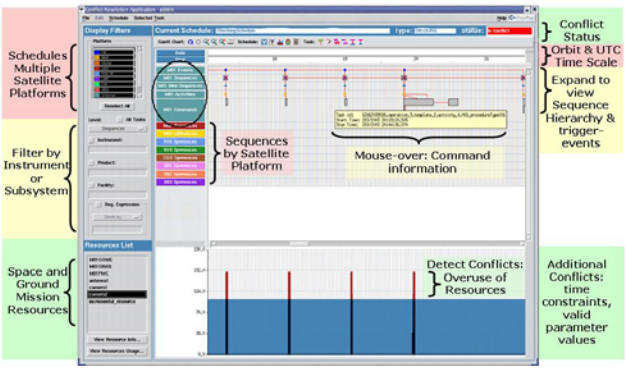

Figure 4. *flexplan* Schedule Generation GUI (SG)

4. Conflict Resolution (CR). In order to obtain a conflict-free schedule, the user must execute this *flexplan* subsystem. It provides tools to detect and resolve conflicts raised by the Schedule Generator which may be due to timeline constraints or over-consumption of resources. Based upon the conflict-free schedule generated by this subsystem, commands can be generated.

In addition, using the Soft Algorithm Generation engine user defined constraint checks can be added to the CR to ensure that mission specific parameters are not violated by the schedule.

5. External Interfaces (EI). All interfaces between *flexplan* and the external ground system elements are managed by this component. It monitors the availability of scheduling products and converts the input products to XML format for ingestion into *flexplan.*

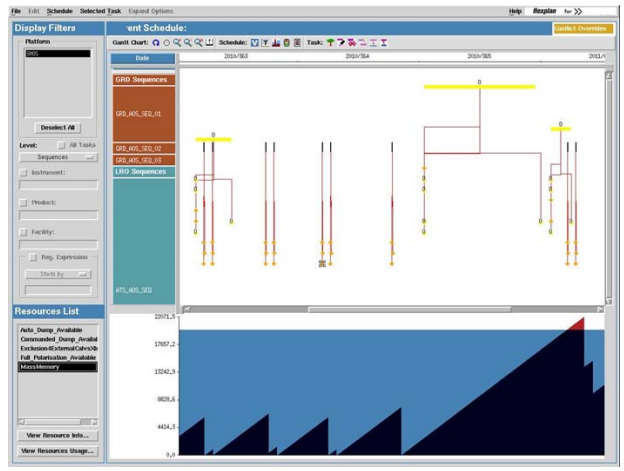

Figure 5. *flexplan* Conflict Resolution GUI (CR)

It has a "pluggable" architecture that uses an XML interface schema (which is publicly available) to adapt easily to different types of missions. Whenever the input products do not comply with the XML schema, adapters are provided in the external interfaces to parse in the various input formats used by the mission.

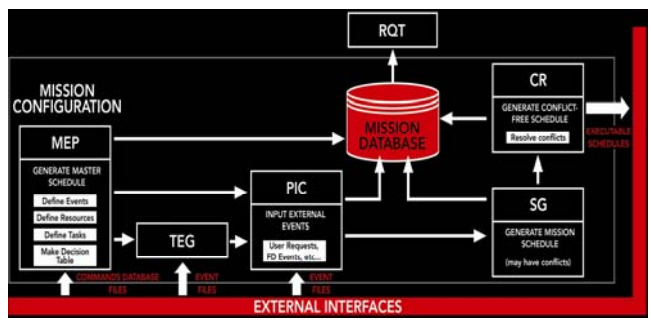

Figure 6. *flexplan* Core Process Architecture

B.Additional Components

The modules described in this section capture the additional planning and scheduling functions that can support missions:

1. Load Builder (LB). The Load Builder module is responsible for converting *flexplan* schedules and tasks into stored command loads for the spacecraft, following the CCSDS tele-command standard. In addition, the load builder will process ephemeris files generated by the Flight Dynamic System (FDS) and convert them into flight software loads.

2. Schedule Execution (SE). The SE is used to visualize the real time execution of the planned tasks scheduled by use of status messages provided by the Real Time System (RTS). The SE can be used to change the schedule in almost real time as well as modify the currently executed schedule based on contingencies.

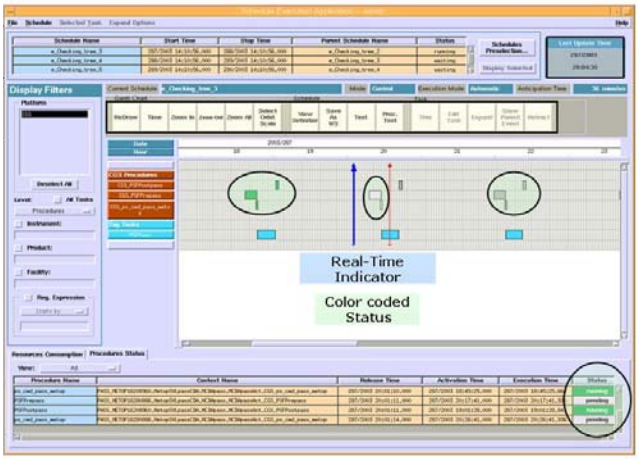

Figure 7. *flexplan* Schedule Execution GUI (SG)

3. Memory Model (MM). The Memory Model module maintains an image of all the files uplinked to the spacecraft and their status as used by the onboard computer. All files in the memory model are archived by category. Its inputs are messages from the RTS, which specifies the name of the memory file and the action performed on that file (e.g. uplink, activate, dump). The output of the Memory Model is a set of reports with the various statuses of the memory files.

4. Activity Plan (AP). The Activity Plan is a web-based client that displays all or a configurable set of scheduled activities and inputs. It allows the user to view execution status of all planned activities based on feedback messages received from the RTS. Several MPS reports such as the onboard stored command load and the ground pass plans are also made available to the user.

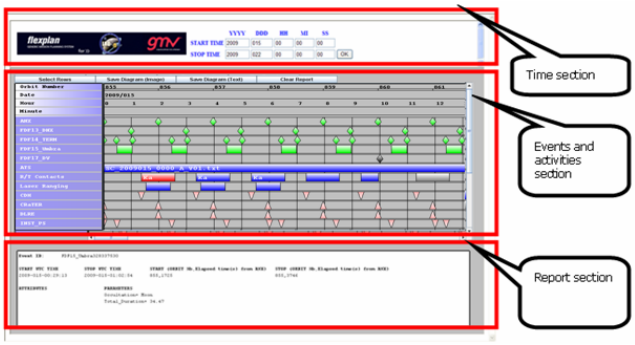

Figure 8. *flexplan* Activity Plan Web Interface (AP)

5. Report Query Tool (RQT). The Report Query Tool allows the user to define MPS reports in ASCII, XML or HTML format, containing any information stored in the *flexplan* database and distribute them via email, hard copy or soft copy.

6. Tailored Events Generator (TEG). The TEG module is used as an extension of the MEP and the PIC to generate custom events specific to a mission constrained to requirements not generally applicable to other missions. It is typically used in missions that require an additional level of planning algorithms for complex planning scenarios.

7. Slew Planning Tool (SPT). This module is similar to the CR module described above, but it implements configurable constraint checks unique to the scheduling of slew maneuvers. The inputs to this module are typically requests from the scientific or earth observation community for slew of an instrument to observe different targets of interest and the output is a conflict-free slew plan provided to the Attitude Ground System (AGS).

A user will typically interact with the MEP when configuring the mission and only rarely thereafter. The daily mission planning loop includes the PIC, SG and CR components. The user will typically start interacting with the PIC by importing the external events (flight dynamics, user requests …), then generate the mission schedule, resolve any conflicts, and finally export the conflict-free schedule to the control center.

III. *flexplan* Strengths

flexplan development has been a change in the approach given to the Mission Planning system concepts in terms of responsibilities sharing. Because of this, communication between Operators and Developers has been reduced to a set of programmatic issues (GUI, interfaces with external systems …). This allows the operators to feel free to implement what they need without having to translate it into requirements specifications.

The following have been found to be particular benefits of *flexplan*:

- The **Soft Algorithm Generation** makes *flexplan* easy to reuse and adapt both during a mission and also potentially expanding it to other missions. Not needing to recompile the system after changes to the mission rules is very highly valued. Significant changes to the mission and flight right up to launch and even during the various phases of the mission have been very easily carried out.

- The **open XML schema** for external interfaces provides a clean, well-documented interface to external entities.

- **Low Cost & Rapid Deployment:** The ability to meet very tight and demanding schedules at a reasonable price was decisive factors in the selection of *flexplan* for missions by Eumetsat, ESA, USGS and NASA. The low amount of specific coding needed for each new mission (typically interfaces, as well as any specific requirements or additional, optional modules), as well as the associated validation, means that costs are kept lower.

- **Multi-mission and Multi-user**: The ability to accept inputs from multiple ground system facilities (and even different ground systems on different continents) and multiple satellites, and integrate them easily has been particularly valued by Eumetsat and NASA. Client-server architecture and multiple access levels provide an additional layer of configurability and versatility to the system.

- **Lower Overhead:** Having a well-documented COTS, which requires a low amount of specific coding for a particular mission,

reduces the managerial overhead both for GMV and the Customer. This is particularly notable in the area of SW modifications, since with *flexplan* they are almost always related to purely SW issues/modifications and not to actual operations. This also results in a fewer change requests and modifications to the system (with all of their implications in schedule and budget).

-**Robustness and reliability**: The *flexplan* core modules have been implemented in all the missions where *flexplan* has been deployed. This means that the *flexplan* core has been largely validated for different scenarios becoming highly stable and reliable. No software anomalies have been identified in the *flexplan* core for the last two years.

IV. Sample *flexplan* Operations

This section describes the Mission Planning and Scheduling process for a Sample Mission using *flexplan*.

Sections A to D describe the scheduling process based on *flexplan*'s modules. Section E is a sample operational scenario. Section F describes how *flexplan* is implemented for automated operations. Section G describes the typical feedback loop between *flexplan* and the Mission Telemetry and Command (T&C) System and finally Section H describes some examples of *flexplan*'s scheduling outputs.

A. *flexplan* Configuration (MEP)

Prior to launch during the mission preparation phase, the Mission Team will use the MEP to configure *flexplan* to support the mission. There are four mission elements to be defined:

- Resources
- Input Events
- Output Tasks
- Master schedules (scheduling rules)

The Resources are configured to keep track of the resource usage and avoid scheduling of conflicting tasks. The Resources can represent physical (e.g. solid state recorder) or logical (e.g. availability of personnel) elements. The user can define the minimum and maximum capacity of each resource as well as whether any use of the resource permanently or temporarily affects its availability.

Planning inputs to be ingested by *flexplan* are defined in the MEP as Events belonging to a configurable category, and having certain parameters and attributes. This gives the Mission Team the flexibility to add new planning inputs to the MPS process or modify the characteristics of existing planning inputs without having to modify the MPS software.

Output tasks are also defined in the MEP as a sequence of subtasks to be place in the schedule and are ultimately the output commands and procedures of *flexplan*.

The tasks are defined under specific platforms to designate them as spacecraft or ground tasks so as to distinguish between them when generating the so spacecraft command sequences and pass scripts. Tasks are made up of four hierarchies: sequences, mini-sequences, activities and commands.

Spacecraft commands and ground procedures are defined in XML database files received from the T&C system and ingested by *flexplan*. Each command can be associated to the use of one or more resources along with the consumption profile.

Finally, master schedules defined as a set of rules that identify conditions based on planning inputs to trigger the scheduling of tasks. The rules can be easily defined and modified by the Mission Team to support various space and ground operations, during the different phases of the mission.

B.Scheduling Inputs (PIC)

flexplan implements a single open XML schema for all planning inputs, of any type. The schema structure provides a flexible XML message that easily maps to any information of the planning inputs. The schema captures the source of the input, the time period for which the input file applies and detailed information regarding each input.

Planning inputs are grouped into different types, each usually corresponding to an input product (i.e. file) in which they are sent to *flexplan*. This grouping allows *flexplan* know which set of products is being updated each time planning inputs are received.

Changes to the scheduling inputs such as updates to forecast products resulting from more accurate orbit knowledge can be received at any time. *flexplan* will use the start and end times of the new scheduling input file to update all of the events belonging to the category of the file being ingested. New events are added, existing events are updated and obsolete events are excluded from the event time space.

In addition, *flexplan* can receive updates and changes to an already received schedule due to last minute changes requested by the Mission Team for upcoming special events, anomalies such as a ground station site going offline or sudden unavailability of a ground station(s) from another spacecraft declaring an emergency. *flexplan* will notify the user of scheduled tasks linked to inputs that are no longer present. Tasks affected by new or updated events are also listed for the user.

C. *flexplan* Schedules (SG)

Scheduling products are delivered to *flexplan* and converted them to XML format. The Mission Team can choose to have the products automatically ingested by *flexplan*, or can manually ingest each product. After the inputs are ingested the Mission Team can modify or delete the inputs events or add new ones via PIC module.

The scheduling rules defined by the Mission Team will identify conditions based on the occurrence or no-occurrence of specific events or requests and will insert tasks in the schedule. These tasks can be defined with default values or use information from any of the

scheduling inputs to further define tasks parameters such as durations, offset, command values etc.

Once the Mission Team is satisfied with the scheduling inputs, the schedule generation begins. There are three types of schedules. They are the Schedule Generation (SG) schedule, the Conflict Resolution (CR) schedule, and the Executable Schedule (ES).

1. SG Schedule. The SG schedule is generated using ILOG Rules to trigger sequences based on specific events. The SG schedule can contain both spacecraft activities and ground activities or the SG schedule can only contain activities related to spacecraft or ground only as needed. In the SG schedule, the Mission Team can manually add, modify or delete sequences as necessary.

2. CR Schedule. After SG schedule completed, a CR schedule is generated from the SG to perform conflict checks. If there are conflicts in the CR schedule, the Mission Team can resolve them or override them.

3. ES Schedule. The last step in completing the daily schedule is to designate it as "Executable". This can be done when the CR schedule does not have any conflicts or conflicts have been overridden. The ES uses the latest events and the latest MEP definition to create a temporary schedule. The temporary schedule is compared with the ES to ensure that the ES is consistent with the latest events and MEP definitions. If there are any discrepancies, the Mission Team receives the warning messages and can use the temporary schedule in the SG to make a new ES. The ES makes a schedule and pass scripts in ASCII format and stores them in a temporary directory. Once the ES is made, the ES can be viewed but cannot be modified.

The ES contains activities for both the spacecraft and for ground operations. Upon exporting the ES, all ground activities are placed into pass scripts for the T&C system, and all spacecraft activities are sent to the Load Builder. The Load Builder converts them to a binary command load for uplink to the Spacecraft. Figure 9 below summarizes the schedule generation process followed by the Mission Team.

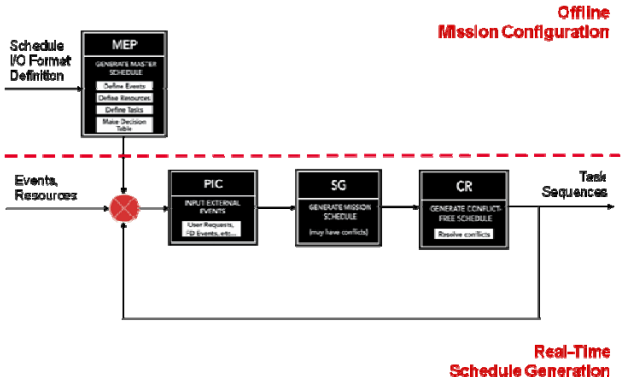

Figure 9. Scheduling Process

D. Conflict Resolution (CR)

flexplan will nominally perform automated operations (see below) where no conflict resolution will be needed other than for Slew Maneuver Requests. The Mission Team will receive off-nadir slew requests from various instrument Science Operations Centers (SOCs). The Slew Planning Tool (SPT) will schedule the various slew requests (off-nadir slew requests or special slew requests) and apply a set of user configurable constraint checks to ensure there are no conflicts.

Once the slew schedule is conflict-free the SPT will export a slew to be sent to AGS to compute quaternion commands. These quaternion commands are then sent back to *flexplan* and ingested as events to trigger slew activities in the daily schedules.

E. Sample Operational Scenario using *flexplan*

This section provides an example operational scenario where *flexplan* provides functionality through each of its components. The Figure below is a summary of an operational scenario. A description of the different steps followed in *flexplan* to produce a schedule for based on this scenario follows.

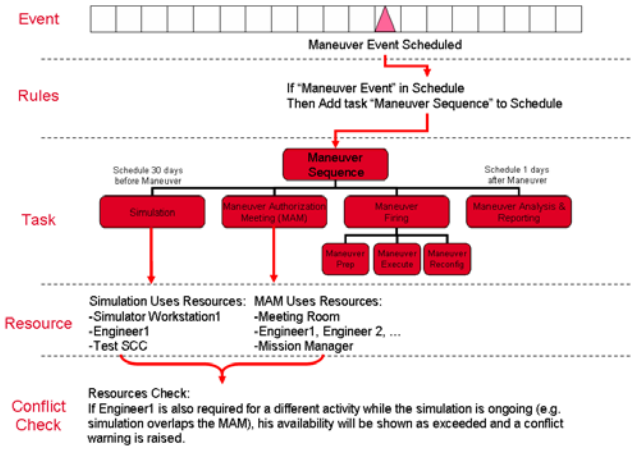

Figure 10. *flexplan* Sample Operational Scenario

Step 1. User defines the mission scenario in the MEP (Mission Environment Preparation module):

- Resources defined: Mission Manager, Engineer-1, Engineer-2, Meeting Room, Simulator, Test Satellite Control Center.
- Events defined: Maneuver request.
- Tasks defined: Maneuver sequence as follows:
  - simulation activity 30 days prior to maneuver
  - maneuver authorization meeting 5 days prior to maneuver
  - maneuver firing activity including:
    - maneuver preparation activities
    - maneuver execution activities
    - maneuver reconfiguration activities
  - maneuver analysis and report generation
- Master Schedule defined: Scheduling rules as follows:
  - **When** there is a maneuver request
  - **Then** schedule maneuver sequence

Step 2. FDS system sends a request to perform a maneuver.

Step 3. *flexplan* ingests the maneuver request in the PIC module (Planning Input Customization module). External data is ingested through the External Interfaces (EI).

Step 4. User generates a schedule in the SG module (Schedule Generation module). *flexplan* will apply mission rules to the input events to add the maneuver sequence to the schedule.

Step 5. User views the schedule in the CR module (Conflict Resolution module) to ensure that there are no conflicts. If any of the resources required (personnel, simulators, etc.) to perform the maneuver activities are unavailable, then the conflicting tasks will be identified and the user must resolve the conflict or override it.

Step 6. User exports the schedule as a set of command procedures and timelines.

F. Automated Operations

Automation of nominal supports is driven by pass scripts generated with *flexplan*. Activities for real-time operations are exported by *flexplan* in the form of pass scripts that conform to formats for example from the Satellite Test and Operations Language (STOL) used by the Mission T&C system. The T&C system will read in the pass scripts using a STOL procedure developed by the Mission Team. The read-in STOL procedure will run in a continuous loop using the network schedule on contacts as a means to identify which pass script needs to be open and read. The pass scripts generated y *flexplan* and the network schedules are the only two input products required to run automated activities. Once the pass script is read successfully, the T&C system will queue each of the scheduled activities as defined in the pass script.

flexplan can be configured by the Mission Team to designate each pass script as “automated” or “manned”. The Mission Team has control of a global setting for all pass scripts but can also specify the automation setting on a per pass script basis. Pass scripts for nominal contacts will generally be automated, whereas pass scripts containing special activities may be configured to be executed manually by the Mission Team. The automatic execution of pass scripts can always be disabled by the Mission Team to continue with manual execution procedures if necessary.

Pass scripts will be used nominally on every contact by the T&C system to execute the planned real-time activities that were scheduled in *flexplan*. The planned activities can either be STOL procedures or Spacecraft commands. Using the pass scripts for manned operations whenever automation is not adequate will cut down on operator error by requiring less typing and real-time decision making in the selection of the proper procedure options. Pass script activities will include everything from start to finish including the procedures to setup the ground system prior to a real-time contact and to clean up the ground system following the contact.

G. The Feedback Loop

In order to update and report the status of all planned activities and of the MPS Memory Model a feedback loop is implemented between *flexplan* and the T&C system. The Mission Team uses a message generation function in the STOL procedures executed by the T&C system to write messages to a post-pass report file detailing the execution status of pass script activates, and also detailing any changes to the onboard memory (e.g. new memory file uplinked) done during the pass.

1. Updating the Activity Map

All Pass scripts will reference the name of the executable *flexplan* schedule from which the pass scripts were created. In addition, all activities in the pass scripts are assigned a task ID included in a feedback message to *flexplan* to identify activity and it execution status.

Each message contains the name of the pass scripts, execution time, procedure unique ID of the procedures and the status. The status of the procedures can be pending, completed or failed. *flexplan* ingests the messages automatically and uses the name of the pass scripts and the unique ID to identify the activity and update its status as indicated in the messages. The Mission Team can then view the status of all planned activities in the Activity Plan.

2. Updating the Memory Model

Each change commanded by the Mission Team to modify the status or content of the onboard memory will generate a message for the Memory Model.

The MPS ingests the messages automatically and updates the status in the Memory Model. The main use of the Memory Model is to determine what was activated in RAM if power-on reset of processor occurs and to track of what load files are loaded and activated.

Prior to launch the Memory Model will be initialized with a file listings done via spacecraft command. Once initialized with launch image, the Memory Model will be updated by ingesting the messages from the feedback log generated by the T&C system after each pass with the Spacecraft.

H. Scheduling Outputs

Sample *flexplan* outputs are spacecraft stored Command Loads and reports, Pass Scripts, Activity Plan, and Memory Model Reports.

1. Spacecraft Stored Command Loads and Reports

flexplan is responsible for generating the two types of stored command loads: Absolute Time Command Sequences (ATS) and Relative Time Command Sequences (RTS). ATS loads will be generated daily and uplinked to the Spacecraft. They are generated by the load builder from a conflict free schedule of activities exported by *flexplan*. RTS loads are defined by the Mission Team using the MEP module of *flexplan*. Whenever necessary these RTS definitions will be exported from *flexplan* and the built into loads by the load builder. Both types of loads will have a

load report detailing the content of the loads and characteristics of the load files.

2. *Pass Scripts*

flexplan generates Pass Scripts used by the Mission Team in conjunction with the T&C system to automate the real-time activities on the ground. Pass Scripts are a listing of all activities, either ground procedure or Spacecraft commands, that get executed by the T&C system. A separate Pass Script is generated for each pass.

3. *Ephemeris Loads and Reports*

The Load Builder module of *flexplan* will also generate ephemeris load files from ephemeris data computed by the Flight Dynamics System. Ephemeris loads will be generated daily and uplinked to the Spacecraft. An ephemeris load report is also generated by the Load Builder including the ephemeris data, and other information regarding number of data point, time step, etc.

4. *Activity Plan*

The AP is a combination of activities from the ATS daily load and the Pass Script, displayed on a web-based timeline. The Mission Team can configure which activities and scheduling inputs are displayed in the Activity Plan. It will be used daily by the Mission Team and external groups to view past, current, and future operational activities and their status as reported by the real-time system feedback messages to *flexplan*. Access to certain MPS reports and input files are also available via the Activity Plan as links associated to timeline elements.

5. *Memory Model Reports*

The Memory Model will track all movements of memory files onboard the spacecraft in order to provide the Mission Team any load file status. Several reports can be generated by the Memory Model containing information about the different categories of load files, the current status of the on-board memory, historical information, etc. The primary purpose of the memory model is to generate a listing of the onboard files and of the active files in the Spacecraft memory space at any given time in the mission, specifically prior to any processor anomalous reset.

V. Conclusion

flexplan is a robust solution for flexible Planning and Scheduling that has supported complex missions for the European Space Agency (ESA) and the European Organization for the Exploitation of Meteorological Satellites (EUMETSAT) and soon the Lunar Reconnaissance Orbiter (LRO) Mission for the National Aeronautics Space Administration (NASA) and LDCM for NASA and the US Geological Survey (USGS).

The use of *flexplan* as a MPS solution validates the use of flight proven COTS in efficient, lower cost ground systems.

flexplan supports any CONOPS for daily scheduling of on-board, real-time and imaging slew planning activities, while allowing the Mission Team to implement modifications to the nominal plan whenever necessary.

As a single point for all mission scheduling, *flexplan* provides the global awareness of the schedule requests and status of activities from the various groups that define the mission daily life.

flexplan allows missions like Eumetsat, LRO or LDCM to automate the repetitive and predictable daily activities as well as to provide a single point of collection for all mission elements contributing to the daily schedule of the spacecraft. It maximizes the time that is available to the Mission Team for ensuring the health and safety of one to multiple spacecraft, ground stations and schedule events accommodating the dynamic needs of the mission and user community.

References

GMV Team, 2009. *flexplan Technical Overview*. GMV Space Systems, Inc.

GMV Team, 2009. *flexplan Generic Architecture Document*. GMV Space Systems, Inc.