FlexPlan: An Operational Mission Planning & Scheduling COTS Used Internationally

J.A. Tejo, M. Pereda, Iker Veiga
GMV S.A.
Calle Isaac Newton, 11
PTM-Tres Cantos
28760 Madrid, SPAIN
+34 91 807 2100
jatejo@gmv.es

J.P. Chamoun, G. Garcia, T. Beech
GMV Space Systems Inc.
1 Research Court, Suite 450
Rockville, MD 20850
+1 (301) 216 3840
jpchamoun@gmvspacesystems.com

Abstract— Every scientific satellite ground segment architecture has its particularities, but they all share certain elements such as the Mission Planning and Scheduling system. This system is highly specific to each mission because of the specific nature and architecture of each mission's Ground Segment, as well as the specific needs of the satellites themselves. Developing a new Mission Planning system from scratch for each mission is a costly endeavor. In order to provide a method for efficiently reusing Mission Planning systems (and thereby, significantly reducing costs), GMV developed a Generic Mission Planning kernel called FlexPlan.

FlexPlan is a highly configurable tool which covers the end-to-end loop of Mission Planning & Scheduling. It allows users to adapt the system to their requirements quickly and easily by using the Soft Algorithm Generator within it. FlexPlan was implemented in such a way that it can be operated by non-experts and can be easily customized to many different types of platforms and missions.

The usefulness of the tool can be seen in the missions and agencies which have selected it. FlexPlan has been selected by three Space Agencies on both sides of the Atlantic for use in a variety of missions. The adaptability of FlexPlan to different mission types is shown by the range of missions which have selected it for their Mission Planning & Scheduling facility. They are LEO, Lunar, single satellite and multi-satellite missions.

EUMETSAT selected FlexPlan for the EPS (European Polar System) program which is part of a joint EUMETSAT and NOAA multi-satellite International Global Precipitation mission. For this LEO mission, FlexPlan handles the mission operations for the Metop satellites, and provides support to the NOAA satellites as well.

ESA selected FlexPlan for the SMOS Planning Generation Facility. SMOS (Soil Moisture Ocean Salinity mission) is a LEO satellite scheduled to be launched by ESA in 2007.

NASA Goddard Space Flight Center selected FlexPlan for the Lunar Reconnaissance Orbiter (LRO) Mission Planning & Scheduling Facility. LRO is NASA Goddard's first Lunar mission and it is scheduled for launch in 2008. As part of the LRO Mission Planning & Scheduling Facility, FlexPlan includes some additional modules for On-Board Memory Management and General Ground Event and Configuration management control.

In a recent upgrade to the tool, a number of modules of have been enhanced, and new features added. These enhancements include:

- A upgraded GUI which provides a clearer graphical representation of planned tasks, as well as an improved representation of conflicts and how to resolve them.
- Planning policies can be based not only upon event triggers (i.e. when a station acquisition occurs, do a data dump), but also upon resource states (i.e. when the SSR fills up, do a data dump)
- A complete XML interface has been implemented and schema defined for the exchange of information with external entities.

This paper will discuss the general SW architecture of FlexPlan, the underlying concepts which it uses, the recent upgrade, current mission implementations, and analyze the benefits and risks of FlexPlan which have been noted by either GMV or the missions which have chosen the tool.

TABLE OF CONTENTS

INTRODUCTION

The development of any satellite mission takes many years, and the ground segment is just one part of this development. Technology is improving in very fast and sometimes dramatic ways. Hardware (computers, peripherals, …) and software can become obsolete very quickly, sometimes in a matter of months.

These two facts create a clear conflict during mission development. When developing a mission, scientists and engineers work with assumptions and equipment which may be upgraded or superseded in only a few months or a year with emerging technologies. These assumptions affect both on-board and on-ground equipment. Some of these technologies must be incorporated into mission development because they may influence the number and type of payload that a satellite is able to carry. This is applicable not only for hardware, but also for software, and it can affect the following aspects of the mission:

- **Instruments:** The on-board instruments may not be the same as initially expected. This may be true not only of the HW used to build the instruments, but also of the very nature of the instruments. Any changes in HW will affect the scenarios used to define the usage of those instruments.

- **Resources:** Like the on-board instruments, the on-board resources may change dramatically from the first steps of the mission to the final load. Resource consumption may be better, batteries may load more energy in less space and may charge in less time, hard disks may be capable of storing more information or even be changed by different devices (e.g. hard disks becoming obsolete), on-board SW may change during the course of the mission.

- **Processing capabilities:** The facilities' processing capabilities may increase due to new hardware development. These capabilities will be present both on-board and on-ground.

- **Satellite:** The satellite itself may change and evolve.

- **Operations scenario:** The scenarios of the satellite operations can be changed in the time span of the mission: new acquisition stations may be added, some communications chains may be changed, instruments may malfunction,…

- **Operations:** Because of all of the changes which may occur to both the HW and SW during mission development and operations, it is clear that the operations will change throughout the mission's life cycle.

All of these HW and SW changes throughout a mission's development and operations affect the ground segment in some way, shape or form. In addition, for scientific satellites, one of the facilities which is part of the Ground Segment is the Mission Planning.

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 ROP is not always a frozen plan. During the lifetime of the mission it will undergo many changes, some quite significant. These changes in the ROP often lead to significant Mission Planning problems.

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. How to develop a cost-efficient Mission Planning system whose main input (the ROP) is not completely known during development, while not running into time-consuming and costly changes in the SW for every change in the ROP?

In order to address this problem, GMV has developed a Generic Mission Planning kernel called **FlexPlan**. This paper will present the salient point and features of **FlexPlan**.

DEFINITIONS

Definitions of some commonly used terms and concepts in this paper are presented here as an aide to the reader.

GUI: *Graphical User Interface*
The GUI is the part of the tool which is apparent to the user and which the user will generally use in order to input information and view output information. It translates all of the user's input and output requests into a format which the **FlexPlan** kernel can use and process, so as to generate the information requested by the user.

ROP: *Reference Operations Plan*
The ROP describes the different scenarios which the mission will encounter during its execution. It also defines the mission and flight rules to be followed throughout the mission.

Resource
From a **FlexPlan** perspective, a resource is anything whose use the Mission Planning system must schedule during the mission. Examples include on-board instruments and equipment, ground station antennas, …

Event
An event is anything which will trigger a sequence of known tasks which depend upon the configuration of the mission and/or flight rules defined in the ROP. Examples of events include flight dynamics events (i.e. ground station signal acquisition or loss, eclipse entry or exit, …), scheduled calibration events (i.e. solid state recorder calibration), and compound events (i.e. a pass over a particular ground station when a simple acquisition or loss of signal is not enough of an identifier). A user request is also considered an Event.

Mission or Flight Rules
Mission or flight rules are those rules which define what command or series of commands will be triggered by a particular event. Examples include what to do during an eclipse, when to down-load data, when to calibrate instruments,…

Scenario
From a **FlexPlan** perspective, a mission is broken up into various scenarios during its lifetime. Each scenario describes a particular phase of the mission. Mission phases may include orbit injection, routine operations, on-route operations (or a dormant period), a change in repeat cycle to coordinate with another spacecraft,… New scenarios may be added during the mission if an instrument dies, ground stations change,… Within **FlexPlan**, each scenario results in a separate Master Schedule which is defined by taking into account the mission and flight rules relevant to each scenario.

Task
A task is a command or series of commands which will be executed based upon the flight rules defined in the ROP when the event which triggers the rule is detected. **FlexPlan** allows up to four layers of nested commands. These four layers are from highest to lowest: sequences, mini-sequences, activities and commands. Commands are the lowest level and are typically imported directly from the Command & Control system. Sequences are made up of mini-sequences, which are in turn made up of activities, which are in turn made up of commands.

BODY OF THE PAPER

When considering the rapid HW and SW changes which happen during mission development and operations, there are two essential points of view:

- The Operator's and
- The Developer's

The operators define the operations of the satellite during its operational life. These operations are the inputs for the work to be done by the ground segment of the mission.

The developers develop the SW which carries out the operations of the satellite defined by the operator. Whenever an operator raises a new requirement which requires a SW change, a long and sometimes difficult communication process between Operators and Developers begins.

Whatever the point of view, one question that arises is clear: Who should be the one to implement new operations, constraint rules, etc. into the Mission Planning System?

The solution provided in **FlexPlan** has been to split the system into two parts by taking changes in operations and rules out of the hardcoded code and putting them into a rules-based editor (the so-called *soft algorithm* explained in more detail below) which can be configured by the operator.. The architecture of **FlexPlan** provides a solution for both operator and developer by allowing a shift in responsibility for implementing changes. Changes can now be implemented by either the developer or the operator.

In the following sections, an overview **FlexPlan** is presented. This includes descriptions of the architecture,

interface, soft algorithms and benefits of **FlexPlan**, as well as a brief analysis of some of the risks.

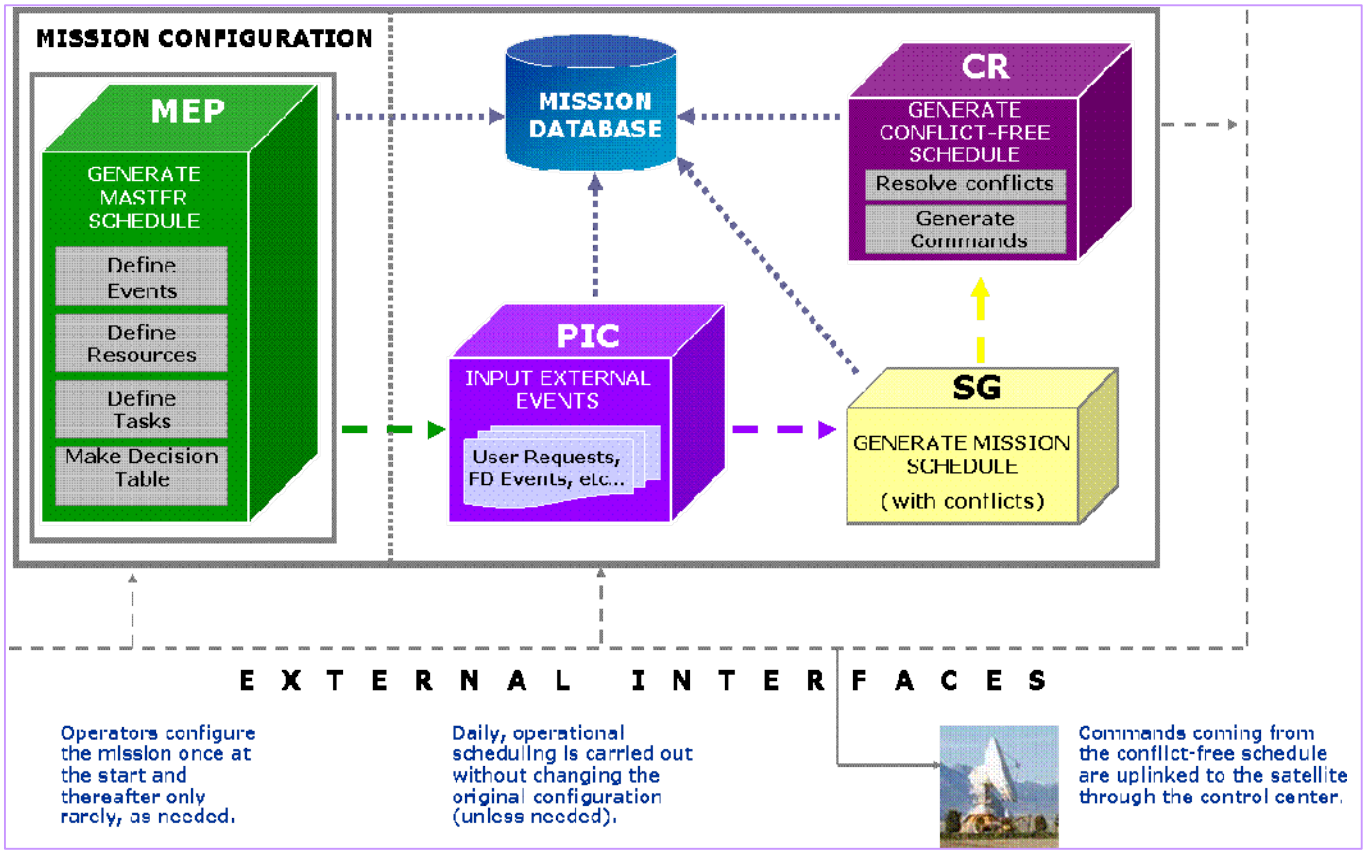

Figure 1: **FlexPlan** Architecture Overview

FlexPlan Architecture

FlexPlan is a highly configurable tool which can be efficiently reused from one mission to the next. It covers the end-to-end loop of Mission Planning and allows users to adapt the system to their requirements quickly and easily. Its features include:

- Soft Algorithm Generation: there is no need to recompile the tool when flight rules change
- Creation of Events and Tasks according to user scenarios
- No programming background necessary to operate the tool

FlexPlan has been implemented in such a way that it can be operated by non-experts and can be easily customized to other platforms and a wide variety of missions.

FlexPlan has a modular architecture in which all components interact with each other via the database. This allows different components to be run at different times or concurrently by different operators. All interaction with outside systems (i.e. Flight Dynamics or Command & Control) is done through the External Interfaces.

FlexPlan consists of five major components:

- **Mission Environment Preparation (MEP):** This off-line 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 typically configured and run at the beginning of a mission. Thereafter, it is run only rarely, as needed, if the mission or flight rules change during the mission. All the elements involved in the mission planning process can be

defined and changed during the operational lifetime of the mission here.

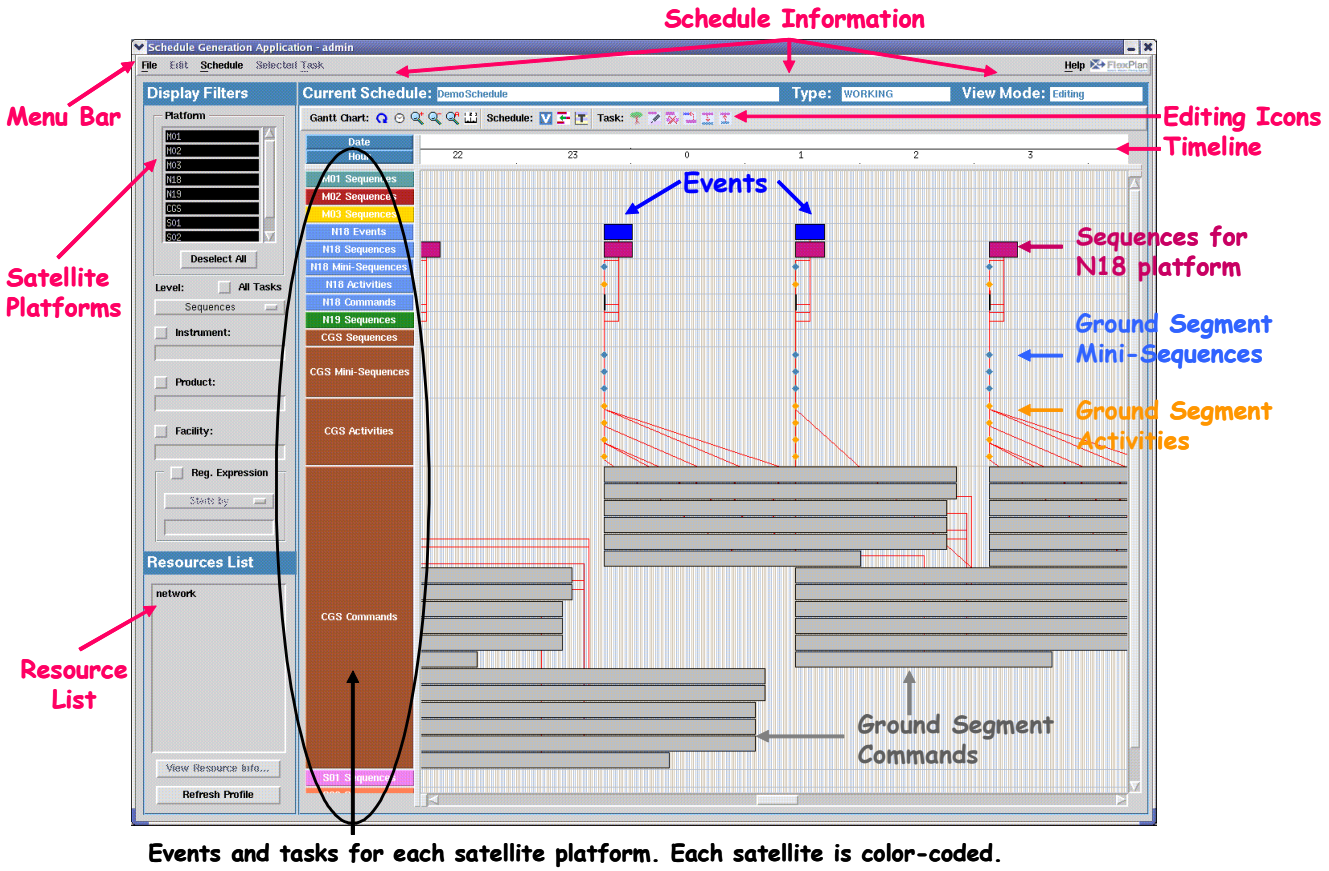

Figure 2: FlexPlan Gantt Chart Screenshot

- **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.

- **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 not be conflict free.

- **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.

- **External Interfaces (EI):** 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.

These five components make-up FlexPlan and provide an easy-to-use, easily customizable solution. The figure above shows the FlexPlan architecture. A user will typically interact with the MEP when configuring the mission and only rarely thereafter. The main 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.

An optional component, the ***Schedule Execution*** (not shown in the figure above), allows the operator to monitor, in real

time, the status and execution of all commands generated by the Mission Planning system within a conflict-free schedule.

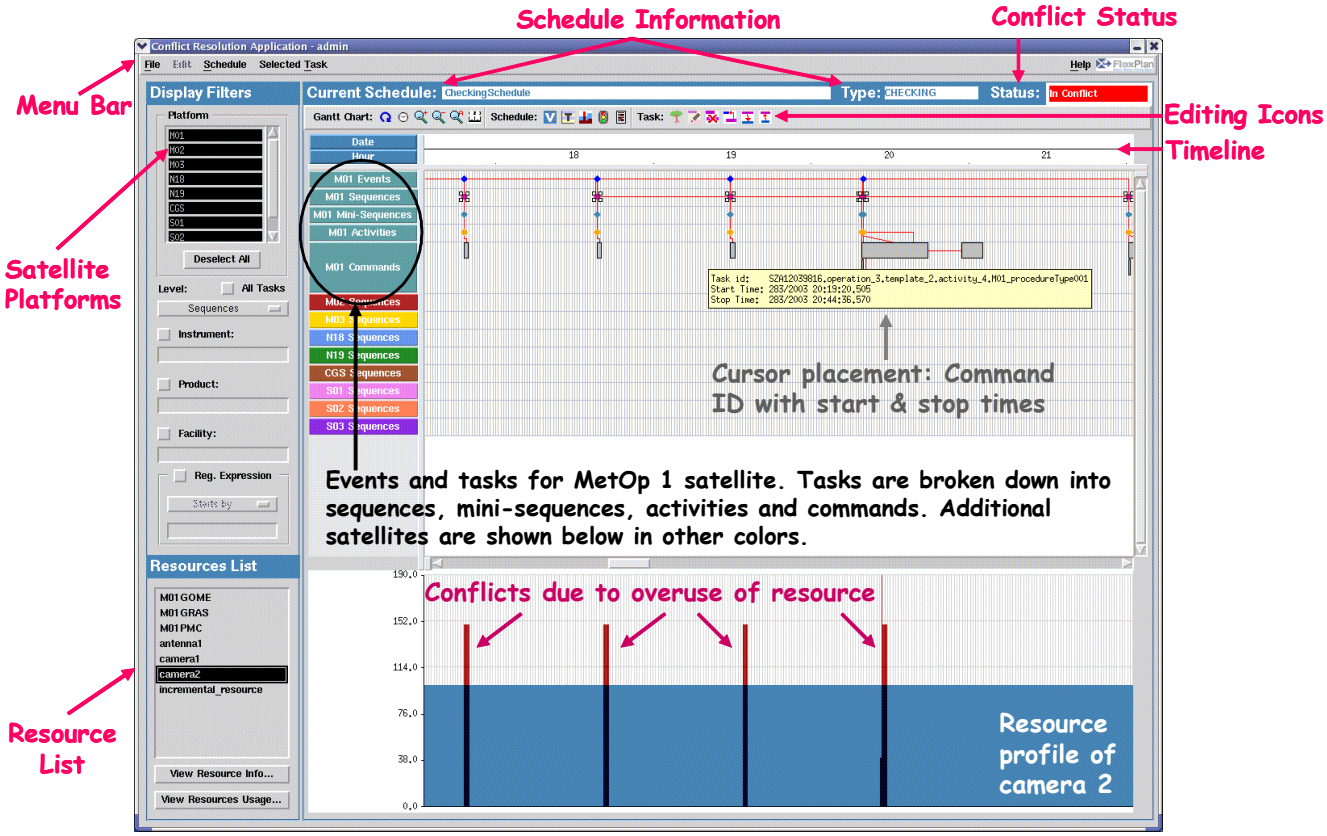

Figure 3: **FlexPlan** Gantt Chart Screenshot showing Conflicts in the Resource Profile

FlexPlan Interface

One fundamental aspect of any SW system is its graphical user interface (GUI). The GUI must be well thought out in order to reduce the possibilities of user error. The screenshot below shows a typical **FlexPlan** Gantt chart interface. In this screenshot, two events (station passes) shown in blue, trigger a series of commands on-board the NOAA 18 satellite and in the ground segment so that a data dump of the solid state recorder may be carried out over the specified ground station.

The main elements of the GUI are as follows:

- **Multi-satellite**: The satellite platform list in the upper left-hand corner shows the satellites whose Mission Planning can be performed. The user may select as many or as few as desired.
- The **Resource List** is in the lower left-hand corner. This list includes all of the resources used during the specified time interval.
- General **schedule information** is along the top.
- The **timeline** is configurable and stretches across the top of the screen.
- Just above the timeline are a series of **editing icons** which allow the user to configure the Gantt chart. The editing can also be done via the Edit menu in the upper right-hand corner, or by double-clicking on the task or event in question.
- **Each satellite is color coded**. All tasks associated with a particular satellite are in the same color. Tasks can be broken down into sequences, mini-sequences, activities and commands. By double clicking on any task, a separate window is activated which provides detailed information on that task.

The Gantt chart GUI is essentially the same for each of the components of **FlexPlan**. The following screenshot shows the Conflict Resolution component. A conflict in Camera 2 usage on-board the MetOp 1 satellite is shown in the **Resource Profile** in the lower third of the screen. The fact that the schedule has conflicts is indicated by the **conflict status** in the upper right-hand corner.

Soft Algorithms

The basis in **FlexPlan** for allowing a shift in the responsibility of implementing changes in the Mission Planning system are the Soft Algorithms included in **FlexPlan**. The Soft Algorithms in **FlexPlan** are a way to allow the user to change rules and resources without having to actually change the code of the system. In this way, no recompiling and retesting of the Mission Planning system are needed. The Soft Algorithms are essentially a scripting language which is used by a Rule Builder to generate the mission and flight rules.

A COTS tool has been incorporated into **FlexPlan** which allows the user to insert soft algorithms and new constraints easily during the mission lifetime. This allows the Operator to update the system during the life of the satellite. New/changed instruments, new/changed ground stations, new/changed operations, etc. can all be introduced anytime during mission development and operations.

ILOG Rules is the COTS which has been used in **FlexPlan**. This tool allows the Operator to include planning rules and algorithms without having to:

- Stop the Mission Planning system
- Recompile and rebuild the Mission Planning system
- Call system administration support, which is something that should never happen, but experience shows that it happens.

The Rules language which is used is not a Programming language in the computer science sense, i.e., it is not like C++ or Java. It is more like a scripting language which can easily be learnt by non-programming specialists. Some training is required, but no specific programming knowledge is required.

The following figure shows an example of a mission rule in **FlexPlan** which uses the Rules language. This rule generates a Calibration operation of an instrument antenna on-board the satellite. It can be read as follows:

When there is an Instrument Calibration event (`INSTR_CAL`) from satellite platform `S01` called `?event`, then the following will happen:

- Create a task whose parent event is ?event and whose name is CalibAftAntennaOpt (calibrate the Aft Antenna)
- Create a task whose parent event is ?event and whose name is S01_CalibAftAntenna (Calibrate Aft Antenna of satellite S01). The task duration will be the triggering event's duration plus 400 ms.

```
WHEN
    there is a Event called ?event
          such that name equals "INSTR_CAL"
          and platform = S01
THEN
    assert Task
          so that parentEvent = ?event.ID
          and name = "CalibAftAntennaOpt"
    assert Task
          so that parentEvent = ?event.ID
          and name = "S01_CalibAftAntenna"
          and duration = ?event.duration + 400
```

Figure 4: Example of Mission Rule using the Rule Builder

The edition of the rules can be a tedious task as well as a source of errors which can be very difficult to debug. In order to edit rules in a user friendly way, the COTS provides a user friendly editor which does not allow syntactically erroneous rules. Once the rules are edited, then the user puts them in a repository which is read by the rules engine which is launched by the **FlexPlan** application.

The following window shows an example of a Rule Builder interface. The panel on the left contains a list of the rules which the user may edit by simply selecting one. The central window is where the selected rule is actually edited. The Rule Builder comes with a pre-defined grammar to help in the definition of the rules. The bottom panel contains all of the grammatical errors which the tool has detected in the selected rule.

So by using this approach, the system has been split into two aspects which are independent of each other in terms of responsibilities:

- The Developer's Side

  All functionalities dealing with purely programmatic issues are left for the developers. These include the GUI issues, communications with other facilities, etc. The operators should not manage these topics. In fact they generally do not have the knowledge to do so (and they do not have to, of course). The only role played in this part of the system by the operators is raising problem reports or new requirements.

- Operators' side

  This is the most interesting part of it. With **FlexPlan**, the operator has been given the possibility of handling directly all issues purely related to operations. These include changes in the

ROP which result in flight rule changes for the Mission Planning system. In the case that operations change, then the operator can AND must perform the changes to the system completely independently from developers and system administrators. Two immediate benefits can be seen from this approach:

- No wasted time in filling change requests, writing specifications, communicating with developers, etc…

The solution to a given problem can be put into the system immediately. There is no delay other than the time needed to write the rule.

FlexPlan Implementations

FlexPlan has been selected by three Space Agencies for use in missions of quite different type and objective.

EUMETSAT selected FlexPlan for the EPS (European Polar System) program which is part of a joint EUMETSAT and NOAA multi-satellite International Global Precipitation mission. For this LEO mission, FlexPlan handles the mission operations for the Metop satellites, and provides support to the NOAA satellites as well.

ESA selected FlexPlan for the SMOS Planning Generation Facility. SMOS (Soil Moisture Ocean Salinity mission) is a LEO satellite scheduled to be launched by ESA in 2007.

NASA Goddard Space Flight Center selected FlexPlan for the Lunar Reconnaissance Orbiter (LRO) Mission Planning & Scheduling Facility. LRO is NASA Goddard's first Lunar mission and it is scheduled for launch in 2008. As part of the LRO Mission Planning & Scheduling Facility, FlexPlan includes some additional modules for On-Board Memory Management and General Ground Event and Configuration management control.

FlexPlan Risks & Benefits

As part of these implementations, GMV has found a number of risks and benefits associated with the use of FlexPlan. GMV has also received feedback from the customers regarding the tool. This feedback is presented here.

FlexPlan Risks

The biggest risk associated with a Mission Planning concept like that used in FlexPlan is the fact that Operators must be trained in the configuration of the rules and perform the configuration management of the rule changes themselves. This implies a certain change of concept for the operator. Some customers choose to have GMV maintain and manage the rule changes, while others have chosen to maintain and manage them themselves.

This change, however, appears to be more than compensated for by the benefits associated with the use of a Mission Planning kernel such as FlexPlan.

FlexPlan Benefits

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 the set of issues related to programmatics (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 were decisive factors in the selection of FlexPlan for missions by ESA 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. A 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).

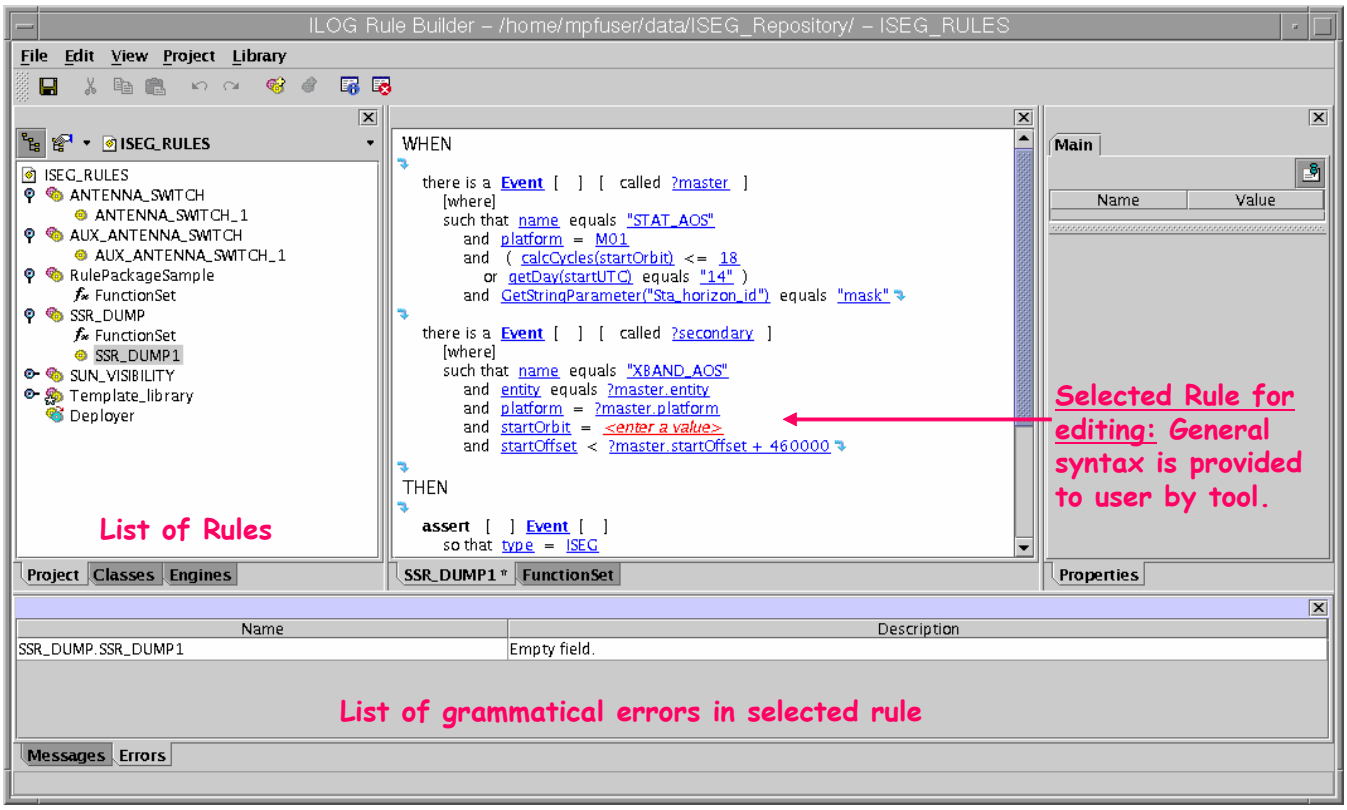

Figure 5: ILOG Rule Builder Screenshot in **FlexPlan**

CONCLUSIONS

Every scientific satellite ground segment architecture has its particularities, but they all share a common architecture which is made up of three main components:

1) the Satellite Control Centre (SCC) for satellite monitoring and controlling via the Telemetry and Telecommand earth terminal (TT&C),

2) the Data Reception Facility which receives data from the Satellite's payload through the X-Band stations, and

3) the Mission Planning and Scheduling (MP) system. The Mission Planning system generates the following items:

- The Payload Work Plan which is the plan to be executed by the Satellite's payload. This is made up of instruments on-board the satellites as well as the Solid State recorders which record the data acquired by the instruments.
- The Uplink Plan which contains the operations to be executed at the TT&C station
- The Downlink Plan, which contains the operations to be executed in the receiving stations.

Although all scientific Ground Segments (GS) share the above-mentioned similarities, it is clear that each MP system is completely different. This is due mainly to the specific nature and architecture of each mission's Ground Segment, as well as the specific needs of the satellites themselves. Developing a new Mission Planning system from scratch for each mission is a costly endeavor. In order to provide a method for efficiently reusing Mission Planning systems (and thereby, significantly reducing costs), GMV developed a Generic Mission Planning kernel called **FlexPlan**.

FlexPlan allows operators (instead of developers) to incorporate directly all changes to the mission planning system which result from changes during the mission development and operations. This change produces significant technical, managerial and financial benefits for Operators who are looking to cut down their development times and costs.

The usefulness of the tool can be seen in the missions and agencies which have selected it. FlexPlan has been selected by three Space Agencies on both sides of the Atlantic for use in a variety of missions. The adaptability of FlexPlan to different mission types is shown by the range of missions which have selected it for their Mission Planning & Scheduling facility. They are LEO, Lunar, single satellite and multi-satellite missions.

REFERENCES