Modeling the Operation of Satellite Payloads for On-Board, Goal-Based Planning

Fabrício de Novaes Kucinskis [1] and Maurício Gonçalves Vieira Ferreira [2]

National Institute for Space Research (INPE), São José dos Campos, Brazil

The concept of goal-based operations is being explored at the Brazilian National Institute for Space Research (INPE), aiming its future missions, through the development of on-board planning software.

After a first prototype for an on-board replanner created for a specific case study, we've started to follow a new path, developing on-board software to be reusable across different missions and aligning our work with the European standards currently in use by INPE. This work culminated in the Goal-based Enabling Software Architecture (GOESA), a multi-mission software capable of performing on-board, goal-based planning.

The core component of GOESA is the Internal State Inference Service (ISIS), an ECSS-like on-board service that contains a model of the satellite's payload. ISIS provides features such as states inference, resources profiling, causal relationships and constraints management, which can be consumed by the ground segment and on-board applications. ISIS also manages the interface between the model and the real equipment, and lets the operations personnel to update its model in flight.

GOESA receives goals (such as "get an image" at a given time) and asks its on-board planner, called LetMeDo, to query the ISIS model in order to explore what-if scenarios, trying to achieve them. The goals can be generated by the ground segment or another flight application. The output of the planning process can be a number of changes on the current operations plan, or even an entirely new plan.

An ISIS model is concise, easy to understand by the engineering team and operations personnel, and light at a runtime perspective – after all, it shall run in space-qualified (and slower) processors. Although light, the modeling language comprises mechanisms that allow a behavioral description close to the real equipment operation.

Up to now, we have exercised the on-board planning for two different Brazilian missions: EQUARS, a scientific mission to study the upper atmosphere, and the remote sensing satellite Amazonia-1. This paper describes the modeling and results gotten, including performance data, from the studies performed for the Amazonia-1 mission case study.

I. Introduction

IN a space mission, the responsibility over the operational tasks is divided between the ground and space segments. The first missions allocated to ground almost all tasks, but the addition of computers to the space segment allowed the operators to start delegating tasks to execution on-board. From then, the continuous increase of the on-board computing power, the experience built up from previous missions, and the requirements of the new ones, have been bringing more and more tasks to the space segment.

Routine procedures such as equipment monitoring and time synchronization are currently performed by the space segment autonomously, to optimize the usage of the communication link and to reduce the effort, and hence the cost, of the operation. But tasks that demand some kind of planning, such as the payloads operation, remain for most missions completely under the responsibility of the ground operations personnel.

Planning consists of the analysis of a number of factors and a reasoning process over which actions to take, and

[1]Technologist, INPE's Aerospace Electronics Division, *fabricio@dea.inpe.br*.

[2]Technologist, INPE's Satellite Control Center, *mauricio@ccs.inpe.br*.

when, to achieve a given mission goal. In a remote sensing mission, for example, the current satellite orbit and attitude, the amount of available storage memory, and the predictions of communication windows are all taken into account, among other variables, to determine when the cameras and data recorder shall be turned on and off. The result of the planning process is a sequence of commands to be sent to the satellite.

The transfer of the planning process, or at least part of it, is something that is being pursued by space agencies in a number of projects (as the ones referred in Section II and also Refs. 1-3). This has the potential to enable new capabilities in future missions, and to better utilize the resources available in current missions.

In this case, the space segment doesn't receive sequences of commands previously defined on ground, receiving instead goals to be achieved. It is up to the flight software, endowed with knowledge about the operations domain and with capability to reason over such knowledge, to perform the planning and determine, on-board, which sequence of commands will make the space segment fulfill the goals received. This is called *'goal-based operation'*.

Although the sequences have been shown to be enough for simple missions, the Brazilian missions foreseen for the next years show a growing level of complexity. In this scenario, the goal-based operation is seen as an evolution of the current paradigm that can meet the operational needs of future missions. The sending of goals to the space segment can lead to a reduction of complexity, effort and cost of operation.

This form of operation can also enable a substantial increase of the autonomy, if the space segment is able to determine by itself, in certain situations, its own goals. This new feature turns the goal-based operation into an autonomous operation of the space segment. A satellite with this capability would be free from the temporal restrictions imposed by the communication with the ground segment, being able to respond in real-time to unpredicted or transitory situations.

Environmental monitoring satellites, for example, could reprogram its activities to get more information about detected floods, fires or thunderstorms in formation, the next time it passes over the event. Scientific satellites would be able to react to short-duration physical phenomena, taking advantage of opportunities that today remain with no response.

The goal-based missions operation and the autonomous operation of the space segment constitute an evolution of the current paradigm of operation through sequences of commands and, as such, bring a number of new problems to be dealt with. Among others, there is the need to determine how to represent, validate and embed the knowledge, how to perform on-board planning with very limited computing power, and how to specify goals in a form that can be handled by the flight software.

The development of technology that enables this kind of operation, however, has the potential to take the space missions operation to a new level, one in which the effort and costs are reduced, at the same time that the usage of the satellite resources increases substantially.

II. Space Missions Operation Paradigms

A. Operation Based on Sequences of Commands

The current operations paradigm for the space segment is based on the generation of fixed and linear sequences of commands, previously defined by the mission operators on ground. Some of these commands shall be executed immediately after received, while others are scheduled for future execution at predetermined moments.

Figure 1 brings a representation of this form of operation. The figure doesn't map accurately the components of any given mission operations architecture, but provide a useful conceptual model for the understanding of the paradigm.

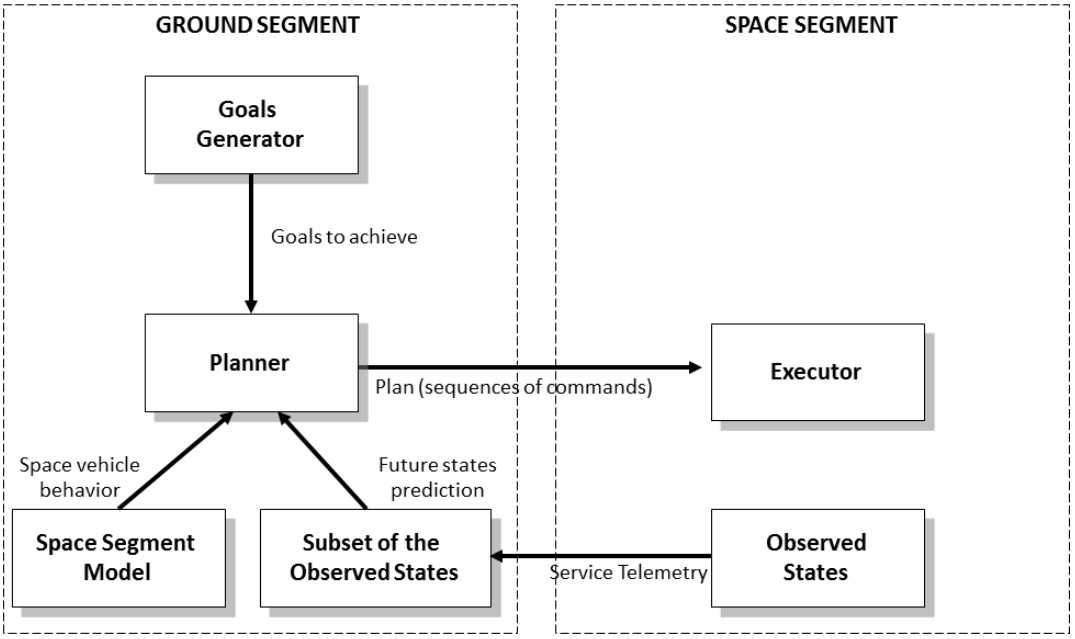

Figure 1. Conceptual model of the operation based on sequences of commands.

In the operation based on sequences of commands, a Goals Generator, usually represented by the mission users, determines the goals to be achieved by the space segment. An example of goal for a remote sensing satellite would be *"acquire high-resolution images of the Admiralty Bay, in Antarctic"* (where the Comandante Ferraz Brazilian Antarctic base is).

It is up to the Planner to turn the received goals into sequences of commands to be sent to the satellite. To accomplish this, it must identify the commands to prepare the camera for the high-resolution imaging, set-up the camera and data recorder, acquire and store the images. After the acquisition, it shall put the equipments again in their non-operational modes.

The execution of these commands must be synchronized with the moment the satellite is passing over the Admiralty Bay, and so the Planner must consult the orbital models to get time references from predicted orbital positions.

During the planning of this operation, the Planner consults the history of past satellite' states, which were transmitted through telemetry, and performs, by using a behavioral model of the space segment, an inference to determine the future states – the ones that are expected to be found at the beginning of the imaging procedure. The result of this planning process is a sequence of commands to be sent to the satellite, each command being associated to a future execution moment.

This form of operation can be performed on ground by automated or non-automated planning. When the planning is automated, the model is explicit and the figure of the Planner is composed by the human operator acting in cooperation with a planning software. When there is no automation, the Planner is the proper human operator, which works with a mental model of the satellite's behavior, based on the mission documentation and its own experience.

The sequence of commands sent to the satellite assumes that the predicted states are correct. Dvorak et al.[4] say that "in order for such command sequences to work as intended, it is essential to accurately predict the state of the spacecraft and its environment at the time of execution of each command". However, it is not possible to always achieve such accuracy.

Due to the limitations on the service communication band, the ground Planner, being automated or not, works with a subset of the satellite' states, registered at greater intervals than those of the on-board sampling. The states information available on ground has a temporal lag that can, for the current Brazilian missions, reach hours.

Besides the difficulty of precisely determine the future behavior of the satellite, it must also be considered the possibility of the occurrence of unpredicted events. On the occurrence of a minor fault or a difference between what was predicted during the planning and what is observed at execution, the sequence of commands will be either aborted or executed as scheduled, but with results different than expected.

In such situation, the whole operation will have been lost. It may take days until the satellite passes again over the same area, and when it does, the information originally needed by the mission user may not be relevant anymore.

The main limitation of the operation through sequences of commands is that they have to be determined on ground, with up to a week of antecedence (in some cases of the Brazilian missions), and have little or no flexibility.

Other forms of commanding are being implemented over the past years, such as the triggering of commands in response to events and the on-board procedures. However, they suffer from the same limitations presented by the 'regular' sequences of commands, since they shall also be planned on ground.

The creation of plans with such advance and the lack of flexibility during the execution force the operators to be conservative and adopt safety margins for the consumption of the on-board available resources, not extracting the most of the satellite.

The trend is that future missions will be increasingly complex and, the higher the operation complexity, the greater the limitations presented by the sequences of commands will be. At some point the sequences may become an operational bottleneck.

B. Goal-Based Operation

Our work adopts the concept of goal as stated by Dvorak et al.[5]: "a goal is a specification of operational intent, meaning that it specifies what we want a system to do". The goal-based missions operation consists of commanding the space segment by telling it *what* has to be done, instead of *how* to do it.

The concept was put in practice for the first time in 1999, when the Remote Agent Experiment (RAX), on-board the National Aeronautics and Space Administration's (NASA) Deep Space One (DS-1) probe generated, at two occasions, detailed operation plans from goals received from ground to perform communication tasks, propulsion tasks and others[6].

Figure 2 shows the conceptual model of the goal-based operation.

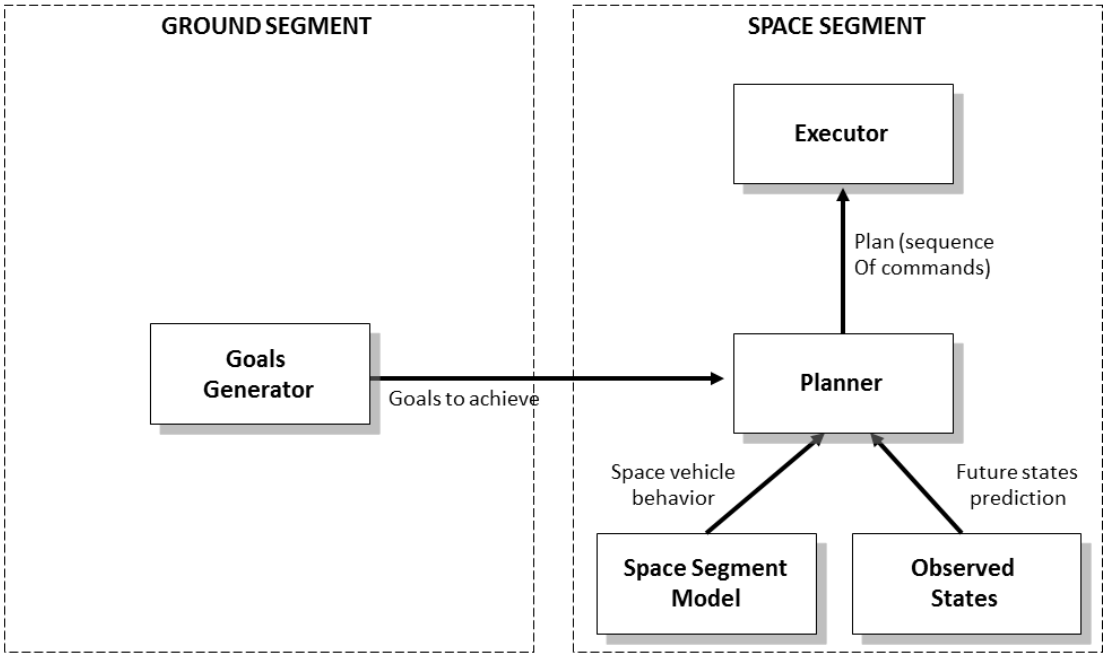

Figure 2. Conceptual model of the goal-based operation.

It can be noted that the components and interactions are basically the same as those present in the operation based on sequences of commands. The main idea of the goal-based operation is the transfer of the planning process to the space segment.

The space segment receives goals (*what to do*) rather than sequences of commands to execute. It is up to the flight software to determine the commands that will lead to the fulfillment of the goals (*how to do*).

Being executed on-board, the Planner has complete and real-time access to the satellite states, which can improve the quality of the future states inference process. Besides, the inferred states can be regularly checked, triggering

corrections in case any deviation is detected between what was predicted and what is being measured. The limitations imposed by the ground planning in advance are thus reduced, as well as the lack of flexibility – since a plan can be adapted on-board in real-time, in response to abnormalities or unpredicted occurrences detected during its execution.

Apart from the DS-1 probe, there are records of only two other space missions that put the goal-based operations concept in practice, both also from NASA.

In the remote sensing satellite Earth Observing One (EO-1), a system named Autonomous Sciencecraft Experiment (ASE) reprogram the operations when it detects events such as volcanoes or floods to get more images in the next orbits. The first in-flight tests of ASE were performed at the end of 2003. Incremental tests occurred since then, until the system was approved for use in nominal operations, in 2005. The EO-1 satellite remains operational to the present date, and ASE keeps receiving updates and optimizations[7].

The third mission to be operated through goals is the Mars Exploration Rovers (MER). Parts of a system called Onboard Autonomous Science Investigation System (OASIS[8,9]) running on-board allows the Opportunity rover to detect and respond autonomously to items of possible scientific interest, such as dust devils and rocks with aspect different from others in a same scene.

The immediate benefits of this paradigm are a greater abstraction on the missions operation, reducing the effort, and hence the cost of operation. There is also a potential improvement on the quality of the generated plans, due to the greater observability of the states.

Another benefit of operating missions through goals is that the execution of on-board planning makes way for the autonomous operation of the space segment, which brings new possibilities to space missions.

C. Autonomous Operation of the Space Segment

To endow the space segment with liberty to decide the best way to achieve goals certainly represents a considerable increase of autonomy – but it is possible to go beyond that.

An on-board planner can provide the space segment capability to autonomously react to failures, unpredicted situations and opportunities, benefiting from its complete and real-time access to the observed states.

For this, the only change in the conceptual model presented in Figure 2 is the addition, to the space segment, of a second Goals Generator. Such change allows for a totally or partially autonomous operation of the space segment. This reduces the dependency of communication with the ground segment in situations that demand for quick responses.

The presence of their own Goals Generator and Planner gives to the space segment a capability to react far superior than that of the current missions. Environmental monitoring satellites could immediately change its plans in order to get more information on detected fires, floods or other events. Scientific satellites could react to short-duration physical phenomena, such as electrical discharges on high atmosphere or the eruption of x-rays from distant galaxies.

It is important to emphasize that the concepts of operation described here are evolutions from each other, therefore not being mutually exclusive. A given mission can adopt any combination of the operations based on sequences of commands, goal-based operations or autonomous operations, according to its necessities.

III. A Software Architecture to Enable the Goal-Based Operations for Brazilian Missions

A first study about on-board planning for INPE's missions began in 2005. The study identified algorithms and modeling techniques which were fit to run in an on-board satellite's computer. This resulted on the development of a prototype for an on-board replanner that had as purpose the change of the original operations plan in order to respond to short-duration physical phenomena detected by a scientific satellite[10].

The performance achieved by the prototype when running in space-qualified hardware, together with its modeling capabilities, encouraged us to start a new project, with a broader vision over the goal-based operations theme. Our new intent was to develop an on-board planning software architecture, reusable across different missions and aligned with the European Committee for Space Standardization (ECSS) standards, currently in use by INPE.

This architecture was named Goal-based Operations Enabling Software Architecture (GOESA). GOESA and its interfaces with the ground segment and other on-board software are depicted in Figure 3.

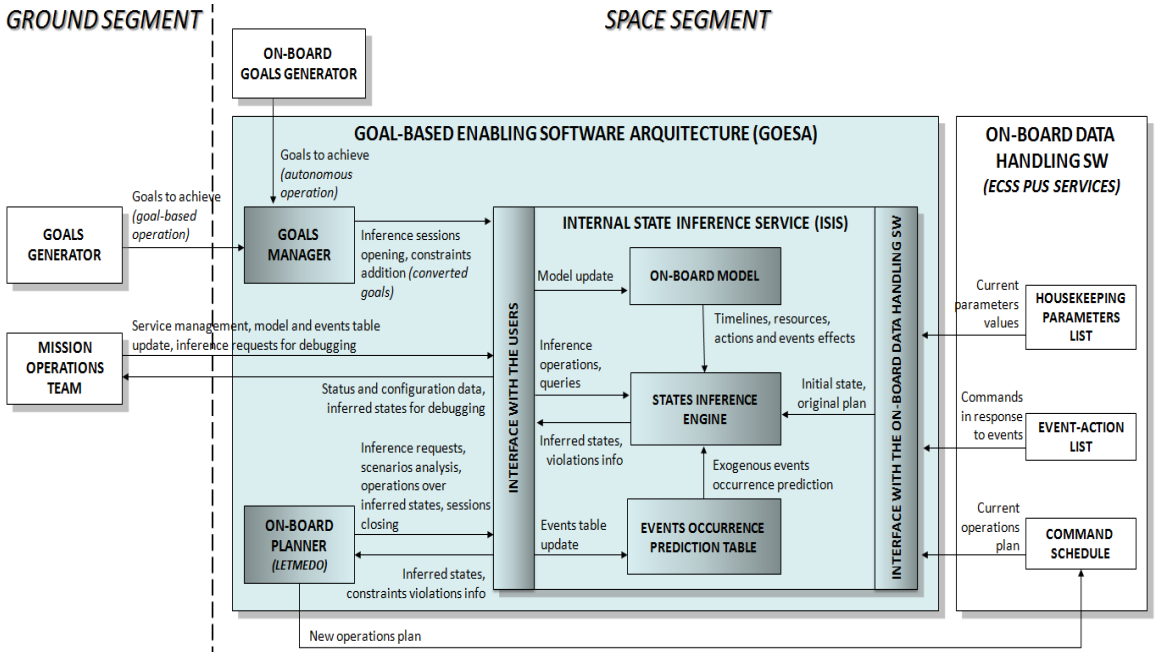

Figure 3. GOESA and its interfaces.

A. Goals Generation and Management

GOESA can receive high-level goals both from the ground segment (in a goal-based operations paradigm) or from a mission-specific On-Board Goals Generator (autonomous operation). An example of an on-board generated goal for a space observatory would be *"point the instruments and acquire more data about the x-ray source that was just detected"*.

Multiple goals can be received at a time. A Goals Manager module verifies the goals' consistency, checks if there are conflicts among them, and converts them in order to be handled by the other modules. The source of the goals (whether they come from ground or from the satellite itself) is not relevant for the rest of the architecture.

B. The Internal States Inference Service

The main component of GOESA is the Internal State Inference Service (ISIS[11]). ISIS is an on-board service that follows the ECSS Packet Utilization Standard (PUS[12]) concepts. It provides features such as states inference, resources profiling, causal relationships and constraints management for other on-board applications, and also for the ground segment (mainly for maintenance and debugging purposes).

ISIS comprises the core modules of the architecture: the On-Board Model and a States Inference Engine, which are described hereafter. It's also part of the service an Events Occurrence Prediction Table, needed to provide to the Inference Engine information about the future behavior of the satellite that is not available elsewhere.

C. The On-Board Model

Systems directed to the achievement of goals by the space segment usually have two of three on-board models, which describe different but interrelated aspects of the same domain. In such systems there is a planning model, a scheduling model, and sometimes an execution or diagnosis model.

To keep the synchronization between models is a difficult and error-prone task. This also opens the possibility for the occurrence of faults generated by small differences between the behaviors predicted by each model.

Indeed, during execution of the RAX experiment, it was detected a fault generated by the existence of multiple models. Verma et al.[13] reported that "one of the two glitches experienced by RAX was due to undocumented and subtle differences in semantics between models in the planning, execution and diagnosis layers".

Differently from those systems, GOESA has a single model of the domain that allows unifying the planning and scheduling tasks, and that also acts as an execution model. This, and the necessity of integration with satellite's hardware and software, demanded the definition of specific modeling concepts and the creation of a new modeling language.

The GOESA on-board model has information on the composition of the payload, its behavior, and additional data for the interface with the satellite that establish the relation between modeled items and their counterparts in the On-Board Data Handling (OBDH) software: data acquired from the equipment and commands sent to them.

D. ISIS Interfaces

In order to perform the states inference on-board, it is necessary to have a correlation between the model elements and the real elements of the domain. This is done by an Interface with the OBDH Software.

Any modeled item shall be related to a piece of information available to the flight software. This relation is established in the description of the domain contained in the model, and it is used to get the current state of the modeled domain, the current operations plan, and to send a new (or updated) plan for execution.

Finally, an Interface with the Users receives and processes requests for changes in the plan, queries to the predicted states, and maintenance tasks as the configuration of service parameters, and the update of the On-Board Model and Events Table.

E. The States Inference Engine

The user goals, the current state and operations plan, and the behavior predicted by the model are all consolidated in a States Inference Engine. This Engine operates over this consolidated information, inferring the future states of the satellite and making the results of such inference available to the Planner.

The Inference Engine allows the Planner to perform 'inference operations' that consist in adding, removing or changing the execution moment of the commands in the operations plan. At each change in the plan, the inferred states are updated. By applying many different inference operations, the On-Board Planner exercises variations of the plan until it finds one that leads to the complete achievement of the received goals.

F. LetMeDo: an On-Board, General Purpose Planner

GOESA On-Board Planner is responsible for, by applying inference operations and queries to ISIS, find the sequence of commands that leads to the achievement of the goals. Planning and scheduling are performed together: the sequence found already considers the resources consumption and the correct execution moment for each command.

No specific planning algorithm is imposed in the architecture. In principle any algorithm could be used, provided that it can run in flight hardware and is adapted to the ISIS interface.

A reference planner was developed. This planner, fondly named LetMeDo, implements a greedy local search algorithm, more specifically a hill-climbing search. It can, however, be configured to behave as a randomized local search. As it is meant to be general-purpose, search heuristics are not used.

Local search is an incomplete method to find the solution for problems, that is, it doesn't guarantee that a solution will be found, if one exists. It is also not an optimal method, since any valid solution ends the search. It is, however, less costly in computational terms than complete methods such as constructive search and frequently shows better results[14].

IV. The Modeling of Domains for GOESA

This Section describes de modeling concepts used to describe the payload operations domain for GOESA.

A. Representation of States Evolution Over Time

GOESA adopts Constraint Satisfaction Problems (CSP) to model the operations domain and hence, to represent the goals. In a domain modeled as a CSP, the states are defined as values of a set of variables, and the goals consist of a set of constraints that such values shall obey.

A property of a CSP is the arity of its constraints. The arity refers to the cardinality, or the size, of the scope of the constraints. While a unary constraint is imposed over a single variable, a binary constraint is imposed over a pair of variables[14], so creating a relationship among them.

In a CSP that has time information, the variables assume not one, but many values over time, all of them respecting the same domain. In GOESA, the time is represented as a variable and is always related to the variables which represent the modeled states through a binary constraint.

We call the related pair of variables 'state X time' a 'timeline'. The timelines are used for the representation of the evolution of variables' values in the modeled system. Resources (as memory, power or thrusters' fuel) are modeled as a special kind of timeline.

The timelines are the basic modeling components in GOESA, over which all other are created. Timelines states are changed by the application of operations over them. The next step, therefore, is to define how to represent the operations.

B. Instantaneous Operations Instead of Fixed-Duration Tasks

The commands executed by the satellite are instantaneous and their effects start to be noted from the moment of execution. In planning systems as RAX and ASE, the commands are usually abstracted into tasks (or activities) with start and end moments. GOESA, instead, models commands as operations with no duration, instead of tasks/activities with duration used in RAX, ASE and others.

An operation is directly related to a command or an event. It changes the states of timelines, which are kept until another operation is applied. An operation can also allocate a fixed quantity of resource, or start the consumption of resources by rates (units consumed per time). Subsequent operations can cease the resources consumption, or start its release also at a given rate. We believe this form of consumption maps more accurately the behavior of the real system than the abstraction of commands to tasks/activities.

C. The Descriptions of the Satellite Operations Domain

The model of the domain is composed by two complimentary descriptions: a Structural Description and a Behavioral Description.

The Structural Description lists the elements of the modeled domain that are relevant for a given application, as well as the timelines and resources that the elements contain. The Behavioral Description has the operations that can change the state of timelines and resources, and describes the effects of each operation.

Both descriptions are unified and turned, by a parser, into C++ source code that can run in INPE's on-board satellite's computers. This source code, when compiled and linked with an ISIS functions library, becomes the On-Board Model.

C.1. The Structural Description

The Structural Description of the model resides in a single file, in XML format. This file specifies:

- The elements that compose the model;
- The timelines and resources that belongs to each element;
- The consumers of the resources;
- The domain of values for timelines and resources;
- The relations between the modeled items and their counterparts in the flight software.

Follows a reduced example of a Structural Description.

```
<structural_description>
<domains>
  <domain name = "CameraModes">
    <value>camera_power_off</value>
    <value>camera_stand_by</value>
    <value>camera_imaging</value>
  </domain>
  <domain name = "RecorderModes">
    <value>recorder_power_off</value>
    <value>recorder_starting_up</value>
    <value>recorder_stand_by</value>
    <value>recorder_record</value>
    <value>recorder_playback</value>
    <value>recorder_erase</value>
  </domain>
  <domain name = "SwitchState">
    <value>real_time</value>
    <value>playback</value>
  </domain>
</domains>
<elements>
  <element name = "Camera">
    <timeline name = "Mode" type="CameraModes" hkparameterid = “10”/>
  </element>
  <element name = "Recorder">
    <timeline name="Mode" type="RecorderModes" hkparameterid = “11”/>
    <timeline name="Ready" type = "bool" hkparameterid = “12”/>
    <timeline name="Switch" type="SwitchState" hkparameterid = “13”/>
    <resource name="Memory" type="uint32" quantity="160000" consumptionperrate="true">
      <consumer name = "Recorder"/>
  </resource>
```

```
  </element>
  <element name = "PowerSubsystem">
    <resource name="Power" type="uint16" quantity="350" consumptionperrate="false">
      <consumer name = "Camera"/>
      <consumer name = "Recorder"/>
    </resource>
  </element>
</elements>
</structural_description>
```

C.2. The Behavioral Description

There are two types of operations that change the states of timelines and resources: actions and events.

An action is a description of the effects of a command executed over timelines and resources. Each action is described by three blocks: preconditions, effects and identification of the corresponding command.

The preconditions are optional and inform which states must be true for the action to be performed. It shall not be possible, for example, to command the record of data if the recorder is not turned on. Logical operators 'and' and 'or' are accepted to state the preconditions. A given action can have multiple preconditions.

The effects describe how the execution of the action affects the timelines and resources.

The identification of the corresponding command is used by the ISIS' Interface with the OBDH Software with two purposes: to extract from the current operations plan the commands that affect the modeled domain, and to send to the flight software, after the planning process, the commands related to the actions of the newly-created plan.

It was created as part of GOESA the ISIS modeling language (ISISml). ISISml have syntax similar to the C programming language, as can be seen in the following example of an action.

```
Action StartRecording
{
    Preconditions
    {
        Recorder.Mode = recorder_stand_by;
        Recorder.Ready = true;
        Recorder.Switch = real_time;
    }

    Effects
    {
        Recorder.Mode = recorder_record;
        Recorder.Consume (Recorder.Memory, 128 per_second);
        Recorder.Consume (PowerSubsystem.Power, 28);
    }

    ServiceRequest
    {
        ServiceType = 134;
        ServiceSubtype = 9;
        DataField = 0A9B8C7D6E5F4; // optional
    }
}
```

The second type of operation, the events, describe the effects of occurrences that affect the states of the model, but that aren't commands. The events can be exogenous or endogenous.

The exogenous events are occurrences out of the control of the flight software, as the entrance and exit of eclipse zones or the beginning and end of the passing over the ground stations.

Endogenous events describe indirect effects of commands. When issuing a turn-on command for a equipment that has a certain start-up time, for example, the moment in which the equipment becomes available for use can be marked by an endogenous event. Another example is the overflow of memory consumption: if a plan has a command to record, but none to stop the recording, an endogenous event can mark the moment when there is no available memory anymore, which automatically interrupts the recording.

Follows an example of the description of an endogenous event in ISISml.

```
EndogenousEvent MemoryFull
{
    RaisesWhen
    {
        Recorder.TotalConsumption >= 160000;
    }

    Effects
    {
        Recorder.CeaseConsumptionOrRelease(Recorder.Memory);
        Recorder.MemoryFull = true;
        Recorder.Mode = recorder_stand_by;
        Recorder.Consume (Recorder.Power, 23); // consumption when in stand-by mode
    }
}
```

Time information for the operations is not specified during the modeling; they will be defined by ISIS, during the states inference process, as a function of the goals received, the current satellite state and the current operations plan.

Being translatable to C++ code, both the actions and events accept conditional and case selection structures, loops and calls to the C/C++ math library.

C.3. Resources Consumption

One of the characteristics that most distinghishes the ISIS modeling from that of other planning and scheduling systems is how it manages the resources consumption.

The Structural Description informs the available quantity of resources and which are the elements that consume them. It shall be also established the form of consumption: if by absolute quantities – as power for example, in Watts – or by rates – as memory, consumed in amount of data per time (e.g. Megabits per second).

Unlike other systems, where the consumption is usually described in fixed amounts for fixed periods related to tasks, ISIS registers *changes in the resources consumption profile for each consumer*.

This means that an operation can indicate the start or end of the consumption of a resource, or a change in the rate in which this resource is consumed or released. It is possible to state, for example, that the action 'Start Recording' make the data recorder start consuming 128 Megabits of memory per second. This consumption will continue until the occurrence of a subsequent 'Stop Recording', or even a new 'Record' action stating a different rate.

V. A Case Study for the Amazonia-1 Satellite

This Section describes one of the case studies applied to the GOESA architecture, for the INPE's remote sensing satellite Amazonia-1.

A. Summary Description of the Satellite's Operation Domain

The Amazonia mission is composed by a remote sensing satellite placed in a polar, Sun-synchronous orbit, at approximately 750 km of altitude. The orbit period is ~100 minutes, with ~35 minutes in eclipse. For this study it was assumed the existence of only one ground station for the reception of mission data, which guarantees 13 minutes of direct communication.

The satellite is equipped with a Charged-Coupled Device (CCD) Camera, a Digital Data Recorder, and a Data Transmitter. They constitute the mission payload.

The acquisition of images when the satellite is out-of-view with the ground station is performed by the execution of time-tagged commands. The Camera and Recorder shall be turned on and put to their operational modes for the image acquisition, and turned off after this.

On the next communication window with the ground station, it is necessary to command the playback of the recorded data, and turn the Transmitter on for the sending of the acquired images. After the transmission of the data, the Recorder memory shall be erased to allow new recordings.

An electronic switch that belongs to the Recorder determines if the data sent to the Transmitter comes from the memory or directly from the Camera. The position of the switch is controlled by commands.

All payload equipment is electrically supplied by the Power Subsystem. In order to use them, one shall first switch their power lines on, and then dispatch the proper 'turn-on' command. Each piece of equipment has its own operation modes. The Data Recorder, in particular, has many modes and configuration commands.

The Power Subsystem is also responsible for the control of the solar panels rotation. As the rotation of the panels cause vibration in the body of the satellite, the panels shall be stopped during the capture of images by the Camera.

The operational goals for the Amazonia mission are always *"acquire and store an image between the moments A*

and B, and send it to the ground", with variations on the number of images to acquire and with respect to whether or not the transmission is part of the goal.

B. The Model Created

Table 1 lists the elements, timelines, resources and domains of values defined for the Structural Description of the model created for the Amazonia-1 case study.

Table 1. Components of the Structural Description of the Amazonia-1 model.

Element	Timeline	Resource	Domain of Values
Power Supply Subsystem	Orbit Period		Sunlight, Eclipse
	Solar Panels		Moving, Stopped
	Camera Line Status		On, Off
	Recorder Line Status		On, Off
	Transmitter Line Status		On, Off
		Power	0 to 350 (0 to 300 when in eclipse)
Camera	Mode		Power Off, Stand-By, Imaging
Recorder	Mode		Power Off, Starting Up, Stand-By, Record, Playback, Erase
	Ready to Operate		True, False
	Memory Full		True, False
	Switch State		Real-Time, Playback
		Memory	0 to 160000
		Data to Download	0 to 160000
Transmitter	Mode		Power Off, Stand-By, Nominal
	Communicating with Ground		True, False

Once defined the Structural Description, the next step is the creation of the Behavioral Description. For this it was necessary to identify in the technical documentation the commands that affect the modeled timelines and resources.

Few simplifications were necessary for the model, with respect to the real commands. Table 2 lists all operations identified for the model.

Table 2. Operations identified for the Amazonia-1 model.

Element	Action or Event Name
Camera	Turn On DC/DC Converters
	Turn Off DC/DC Converters
Digital Data Recorder	Turn On DC/DC Converter
	Turn Off DC/DC Converter
	Playback
	Erase File
	Stop Playback
	Start Recording
	Stop Recording
	Set Switch to Playback
	Set Switch to Real-Time
Data Transmitter	Transmitter-Camera Channel Turn On
	Transmitter-Camera Channel Turn Off
Power Supply Subsystem	Switch On Camera Line
	Switch Off Camera Line
	Switch On Data Recorder Line
	Switch Off Data Recorder Line
	Switch On Data Transmitter Line
	Switch Off Data Transmitter Line
	Stop Solar Panels
	Move Solar Panels

Element	Action or Event Name
(Exogenous Events)	Enter in Eclipse
	Exit Eclipse
	Start Communication
	Finish Communication
(Endogenous Events)	Memory Full
	Memory Empty
	Data Recorder Ready

C. Scenarios and Performance of GOESA for the Amazonia-1 Goal-Based Operations

Various scenarios were exercised for the goal-based operations of the Amazonia-1 satellite, with variations on the initial states, the original operations plan and the goals to achieve.

Table 3 summarizes the average planning time for the scenarios, grouped by its characteristics. The times were registered with GOESA running in a prototype for an on-board satellites' computer, which has an ERC32 processor (SPARC V7 architecture) running at 15 MHz (5.8 MIPS). The allocation of processing time was 100%.

Table 3. GOESA performance in scenarios for the Amazonia-1 mission.

Characteristics of the scenarios	Planning time
Creation of a new operations plan to acquire and store an image.	Between 1 and 3 minutes
Change of a preexisting operations plan to acquire and store a new image, with few impact on the original plan.	Between 1 and 3 minutes
Change of a preexisting operations plan to acquire and store a new image, and to send all recorded data to the ground, with great impact on the original plan (new acquisition comes before what was the originally scheduled).	~ 20 minutes

The planning time for scenarios of the last type, considered the worst case, would allow GOESA to generate a response for a received goal in less than 100 minutes, using 25% of the processor time. This means that if a plan were asked on a given orbit, it would be available for execution on the next one.

Although the developed planner is not proposed to have optimal performance, the times registered are comparable to those reported by the projects RAX[1], ASE[15] and OASIS[16], which run over similar hardware.

The future INPE missions will use more powerful processors. We believe that, with such processors and an eventual optimization of the planner and ISIS, the planning time for the worst cases could be reduced to 10 or 20 minutes, with 20-25% of processor usage.

Compiled, GOESA needs only 56 kbytes of program memory (including 9 kbytes for the Amazonia-1 model) and less than 200 kbytes of data memory during the execution of the on-board planning sessions.

VI. Final Remarks

It was conceived and developed a software architecture for on-board planning, capable of enabling the goal-based space missions operation paradigm, and consequently increase the autonomy of the space segment.

The performance presented by the architecture in a realistic case study is comparable to the ones reported by the three NASA missions that have run on-board planning in the space segment up to date.

A new planning modeling language was created, that allows a description of the operations domain closer to the real system and with little abstraction.

The next step for this project is to work more on the integration with the OBDH software in order to allow GOESA to be considered for a technological validation experiment in a future Brazilian mission.

Acknowledgements

The authors would like to thank the Advanced Inertial Systems (SIA) project and the Research and Projects Financing (FUNDEP) for the financial support.

References

[1]Muscettola, N., Dorais, G. A., Fry, C., Levinson, R., Plaunt, C., "IDEA: Planning at the Core of Autonomous Reactive Agents", *6th International Conference on AI Planning and Scheduling (AIPS)*, Toulouse, France, AIAA Press, 2002. p. 58-60.

[2]Woods, M., Shaw, A., Barnes, D., Price, D., Long, D., Pullan, D., "Autonomous Science for an ExoMars Rover-Like Mission", *Journal of Field Robotics*, v. 26, n. 4, p. 358-390, 2009.

[3]Beaumet, G., Verfaillie, G., Charmeau, M. C., "Feasibility of Autonomous Decision Making On-board an Agile Earth-Observing Satellite," *Computational Intelligence*, v. 27, n. 1, p. 123-139, 2011.

[4]Dvorak, D. L., Amador A. V. and Starbird, T. W., "Comparison of Goal-Based Operations and Command Sequencing", *Proceedings of the 10th International Conference on Space Operations*, Heidelberg, Germany : American Institute of Aeronautics and Astronautics (AIAA), 2008.

[5]Dvorak. D. L., Ingham, M. D., Morris, R. M. and Gersh, J., "Goal-Based Operations: An Overview", *Proceedings of the AIAA Infotech@Aerospace 2007 Conference and Exhibit*. Rohnert Park, CA, USA : American Institute of Aeronautics and Astronautics (AIAA), 2007.

[6]Jónsson, A., Morris, P., Muscettola, N. and Rajan, K., "Planning in Interplanetary Space: Theory and Practice", *Proceedings of the Internacional Conference on AI Planning and Scheduling*, Breckenridge, CO, USA. pp. 177-186, 2000.

[7]Chien, S., Tran, D., Rabideau, G., Schaffer, S., Mandl, D. and Frye, S., "Improving the Operations of the Earth Observing One Mission via Automated Mission Planning", *Proceedings of the 11th International Conference on Space Operations*, Huntsville, AL, USA : American Institute of Aeronautics and Astronautics (AIAA), 2010.

[8]Estlin, T., Castaño, R., Gaines, D., Bornstein, B., Judd, M. and Anderson, R. C., "Enabling Autonomous Science for a Mars Rover", *Proceedings of the 10th International Conference on Space Operations*, Heidelberg, Germany : American Institute of Aeronautics and Astronautics (AIAA), 2008.

[9]Jet Propulsion Laboratory (JPL), "Under their own AEGIS", *JPL's Science and Technology News, 2009*: http://scienceandtechnology.jpl.nasa.gov/newsandevents/newsdetails/?NewsID=677 [cited 27 February 2011].

[10]Kucinskis, F. N. and Ferreira, M. G. V., "Dynamic Allocation of Resources to Improve Scientific Return with Onboard Automated Replanning", *Space Operations: Mission Management, Technologies, and Current Applications*, Progress in Astronautics and Aeronautics Series, v. 220, Chapter 20, pp. 345-359, AIAA, September 2007.

[11]Kucinskis, F. N. and Ferreira, M. G. V., "Taking the European Committee for Space Standardization Autonomy Concepts One Step Further", *Space Operations: Exploration, Scientific Utilization, and Technology Development*. Progress in Astronautics and Aeronautics Series. Reston, VA, USA : American Institute of Aeronautics and Astronautics (AIAA), v. p. 187-199.

[12] European Committee for Space Standardization (ECSS), "Ground Systems and Operations - Telemetry and Telecommand Packet Utilization", ECSS-E-70-41A, ESA Publications, January 2003.

[13]Verma, V., Jónsson, A., Estlin, T. and Levinson, R., "Survey of Command Execution Systems for NASA Spacecraft and Robots", *Proceedings of the International Conference on Automated Planning and Scheduling*, Monterey, CA, USA : American Institute of Aeronautics and Astronautics (AIAA), pp. 92-99, 2005.

[14]Dechter, R. "Constraint Processing", 1st. ed, *The Morgan Kaufmann Series in Artificial Intelligence*, Waltham: Morgan Kaufmann Publishers. 481 p. ISBN-13 (978-1558608900), 2003.

[15]Tran, D., Chien, S., Rabideau, G. and Cichy, B., "Flight Software Issues in Onboard Automated Planning - Lessons Learned on EO-1", *International Workshop on Planning and Scheduling for Space*, Darmstadt, Germany, Darmstadt: STCI, 2004.

[16]Bornstein, B., Estlin, T., Gaines, D., Thompson, D. R., Granville, C., Burl, M., Castaño, R., Anderson, R. C., Judd, M. and Chien, S., "Engineering AEGIS Autonomous Science Targeting for the Mars Exploration Rovers", *Workshop on Spacecraft Flight Software*, Pasadena, CA, USA, 2010.