SAC-D/Aquarius Mission Automated Plan Generation and Execution Validation

Eduardo Romero, Marcelo Oglietti, Estefanía De Elía
CONAE - Argentine National Space Agency
CETT, RP C45 Km 8, Córdoba
Argentina
{eromero}{marcelo.oglietti}{edeelia}@conae.gov.ar

Abstract.
SAC-D/Aquarius is a joint satellite space mission with CONAE and NASA as main partners. For this mission, a distributed planning approach has been operational since the spacecraft launch on June 10, 2011.

In this paper we describe the representation used by the distributed planning system that runs at the Mission Operations Center. We describe how this representation is exploited for automatic generation of all activities related with periodic orbital events and for an automatic verification of a successful plan execution. The former takes advantage of an abstraction layer provided by the planning system and the latter is based on the model inside the planning system that connects all activities considered when planning with their detailed implementation.

1 Introduction

SAC-D/Aquarius is a low-orbit earth-observing science satellite launched on June 10, 2011 on a Delta II launcher at Vandenberg NASA facilities. The orbit is sun-synchronous quasi-polar at 657 km altitude. The revisit period of the orbit is seven days and has been maintained flawlessly by the orbit dynamics team by performing the usual orbit correction maneuvers.

The satellite includes eight instruments and, as usual with Earth Observation Satellites, its design is based on the assumption that all science operations commands plus some satellite maintenance commands are generated and upload from ground. The on-board software autonomy mainly focuses on executing power, thermal and attitude maintenance; plus fault detection isolation and recovery automatisms. All data downloads, science sensor acquisitions plus some maintenance activities –like orbital maneuvers– are planned from ground. Since launch, on-board operations have been managed with the same planning system, including all non routine launch, early orbit, deployment and commissioning activities.

The distribution of the satellite service platform resources between all instruments is defined in several fixed budgets: on-board data storage, time-tagged command storage, power, etc. As we explain below, these budgets allow the implementation of a planning scheme that up to a certain degree is regular and that can be automatically generated starting from a definition of high level science targets.

The satellite operations plan is made in a distributed manner from three contributions. Both (a) the Aquarius Instrument Operations Team and (b) the SAC-D Instruments Operations Team generate a set of action requests that are later combined with those generated by (c) the Flight Operations Team –covering all satellite service platform and coordination activities–. These actions requests are the starting point for the detailed planning of the satellite that results in a series of detailed scripts that will be used for handling each contact, containing all the commands to be uploaded plus real time operational procedures if needed, see (Romero and Oglietti 2009) for more details.

An automatic system for the generation of the contribution by the Aquarius instrument operations team (case a) has been in place from launch, but due to the project schedule and some other constraints, we left out of the first deployed version the following two functions that were originally envisaged for the planning system:

- To automatically generate all regular activities contributed by the SAC-D instrument operations team (case c) and the Flight Operation Team (case d)–, starting from a high level science target definitions and from certain operational rules, and maintaining all the flexibility needed for the mission;
- To monitor the plan execution and compare the predicted behavior with the values observed in the telemetry downloaded in order to generate alarms/warnings that cannot be computed by the usual check of limits alarms.

Since for SAC-D/Aquarius mission, high level science goals are fix for considerable long periods that can be measured in months, the associated acquisition activities for each science goal can be automatically computed from the satellite accesses to the corresponding targets. Furthermore, given that the mission has a seven day revisit period, these accesses follow a periodic pattern of seven days that can be used to simplify the computations.

As it is usual in spacecraft operations, we define alarm limits over several hundreds of variables and check them

for every downloaded frame (one every eight seconds) in order to detect any serious anomalies. The referred functionality shall monitor the execution of the plan by comparing the predicted behavior with the values observed in the downloaded stored telemetry.

The absence of these functionalities entailed more manual and error prone work for the flight operations team.

In this paper we describe the design and implementation of these two new functionalities of the planning system. The first is already operational and has relieved a lot the workload of the SAC-D Instruments Operations Team and the Flight Operations Team (from days to hours). The second is in the final validation and deployment phase and it will add early detection features about unexpected behaviors that otherwise would not trigger any computed on board system alarm. For example, if a command fails to act properly, in general, this does not imply that we will see an alarm, because alarms are associated with variables off-limit values, and they might be right even after the command failure. If this happens, using only a variables off-limit check fails to capture the problem.

1.1 Plan Generation

After the commissioning phase, when the mission operations turned to nominal phase, science plans become more stable. At that point several science targets were fixed as periodic activities to be carried out on a weekly base given that the periodicity of the orbit is seven days. This helped the deployment of the new system to automate the generation of the SAC-D instruments request and that of the operations plan. We keep the manual generation approach because we follow a mixed initiative approach and also because in this manner it is possible to add exceptional activities when needed.

Furthermore, regarding science data takes, five of the eight SAC-D/Aquarius instruments generate science data in a continuous and uniform rate (including the main instrument Aquarius). Therefore, for them, only the data downloads need to be addressed. Instead, the acquisition activities of the other three instruments need to be specifically planned. These instruments are: an infra-red camera (NIRST), a high sensitivity camera (HSC), and a data collector system (DCS).

It is important to notice that there are not to many examples of automatic *science goal generation* in the literature. Most efforts are put in how to construct an optimal valid plan from an oversubscribed set of science targets. But science targets are usually inputs for the planning systems, and are collected from mission users or scientists.

A remarkable example of automatic science target generation is the EO-1 mission (Chien *et al.* 2003), where the science targets are automatically generated on-board by the identification of valuable data take opportunities. Another interesting example is SensorWeb 2.0 presented in (Mandl *et al.* 2008). This paper presents an ambitious space sensor web for disaster management with the objective of facilitate the United States contribution to the Global Earth Observation System of Systems (GEOSS). GEOSS is a worldwide initiative in this direction, with the objective to form a network of EOSs for a wide range of applications in order to provide a real-time picture of the whole planet by sharing all countries sensor resources. This sensor web relays on most important standards in the area like the Open Geospatial Consortium (OGC) and the Sensor Web Enablement (SWE) suite. SensorWeb 2.0 intents to present to the user the most simple possible experience integrating automatically several space, air and ground sensors, e.g. Moderate Resolution Imaging Spectrometer (MODIS), NASA's Earth Observing One (EO-1), the US's Air Force Weather Agency and an Unmanned Aerial System (UAS). The sensor web allows the users to define their regions of interests and then the system automatically detects events of interest. What the users wants to see is automatically executed by means of an appropriate workflow and the best available sensors. For example, if a fire is detected by inspecting MODIS data, this automatically triggers a higher resolution instrument like the Hyperion on the EO-1 satellite to take a higher resolution image, which in turn also automatically triggers an Unmanned Aerial System take for more detailed imagery.

In our work, the planning process is made at ground like in SensorWeb 2.0. We consider orbit events –like ground station contacts and target accesses– plus the definition of high level science objectives to produce the plan. The definition of the high level science objectives is done by means of specifying the areas of interest plus some critical parameters like for example the frequency with which the images are needed and a seasonal validity period for the area outside which the target has no scientific interest. Taking advantage of an abstraction layer provided by the planning system, this system verify the plan by means of a model of the whole satellite operations, and after a successful verification, it generates the detailed plan of commands to be upload to the spacecraft.

1.2 Plan Execution Validation

In the opposite direction, the automated verification of plan execution is done by using the model of the planning system that connects the activities with their detailed implementation in terms of commands. By matching the command execution effects, modeled in the planning system, with the telemetry processed by this application, the system is able to automatically verify the post-facto correct execution of the plan, even for activities that would not raise a system off-limit check alarm if not executed.

The rest of the paper is structured as follows. In order to give a self-contained presentation, we first present a review of the representation used in the planning system.

Then we explain how the automated plan generation was designed and implemented. After that, we present the plan execution validation design. Following, a few conclusions about present work.

2 The SAC-D/Aquarius Mission Planning System

In this section we review the planning system. We start by introducing the mission objectives. Then we present the main concepts and data structures that are used in the planning system to build the plan. For more details on this matter, we refer the reader to (Romero and Oglietti 2009).

2.1 SAC-D/Aquarius Mission Overview

The SAC-D/Aquarius Mission is an Earth Observation Satellite cooperative mission. The main partners are the Argentine national space agency Comisión Nacional de Actividades Espaciales (CONAE) and the United States National Aeronautics and Space Administration (NASA). Also, ASI –the Italian National Space Agency– and CNES –the French National Space Agency– participate in the mission with two science instruments.

The primary science objective of the mission is to contribute to the understanding of the whole Earth system including the effects of natural and human-induced changes on the global environment by measuring sea surface salinity and other variables that aim to resolve missing physical processes that link the water cycle, the climate, and the ocean. SAC-D/Aquarius observatory has other instruments that complement and help to improve overall performance. The SAC-D/Aquarius instruments are the following:

1. Aquarius;
2. Microwave Radiometer (MWR);
3. New Infra-Red Sensor Technology (NIRST);
4. High Sensitivity Camera (HSC);
5. Data Collection System (DCS);
6. Radio Occultation Sounder for Atmosphere (ROSA);
7. Cosmic radiation effects and orbital debris and micrometeroids detector (CARMEN-1);
8. Technological Demonstration Package (TDP).

The Aquarius instrument was provided by NASA, MWR, NIRST, HSC, DCS and TDP were provided by CONAE, CARMEN-1 instrument by CNES and ROSA instrument by ASI.

These instruments complement each other and have the following multiple objectives: measurement of sea surface salinity, measurement of rain rates, surface wind speeds, water vapor and cloud liquid water over the ocean, hot spots, high temperature events and volcanic eruptions, sea surface temperature, temperature and humidity profile of the troposphere and the stratosphere, light intensity over urban areas and polar auroras, hot spots mapping of fire risk, soil moisture, etc.

The architecture of the satellite follows the usual modular design ((Larson and Wertz 1999; Boden and Larson 1996)), where there are several dedicated service platform subsystems (Command, Control & Data Handling, Power, Attitude, etc) that provide services to the payloads.

The mission has the support of several ground stations. For nominal operations it uses CONAE's ETC (Estación Terrena Córdoba, located in Argentina) and ASI's Matera GS (located in Italy). ETC provides both X-band an S-band services, and Matera provides X-band services. For the Launch and Early Orbit Phase (LEOP), and for maneuvers and contingencies, it also has the support of NASA's Near Earth Network (NEN) with four ground stations located around the globe (Wallops Isld.-USA, McMurdo-Antarctica, Fairbanks-Alaska and Svaldbard-Norway) that provides S-band services. Also for LEOP the mission had the support of ASI's Malindi GS, in Kenia.

2.2 Planning Problem Representation

The planning system is built on concepts very similar to those presented in (Dvorák and Barták 2010). In this section we explain how the planning problem is represented and handle in the SOP (Spacecraft Operation Planning) subsystem at the mission Mission Operations Center (MOC).

State Variables At the MOC, the satellite and some other aspects of the mission are modeled as an evolving set of variables called, *State Variables* (SV). These State Variables are used to represent in a succinct way the important aspects of the spacecraft that are necessary to consider for planning.

The use of SVs and timelines in this way is widely extended as a *de facto* standard, as evidenced for example in RAX-PS or ASPEN systems (Jónsson *et al.* 2000), (Chien *et al.* 2000).

Each satellite instrument and subsystem is modeled with a set of SVs that represent the operational modes and all the important information that summarize the state of the instrument or subsystem.

There are also a few SVs that are used to represent information external to the spacecraft. These kind of SVs are called None Controllable State Variables (NCSV). For instance, there are NCSVs indicating eclipses, propagated and scheduled ground station contacts, etc.

We restrict the specification to provide a way of controlling the important aspects during nominal operations. And because the plan is made in a distributed way – as explained above the Aquarius Instrument Operations Team adds to the plan all Aquarius instrument activities– the state variables are also used to allow the proper synchronization of the activities added by each team.

Formally, a State Variable is just an identifier together

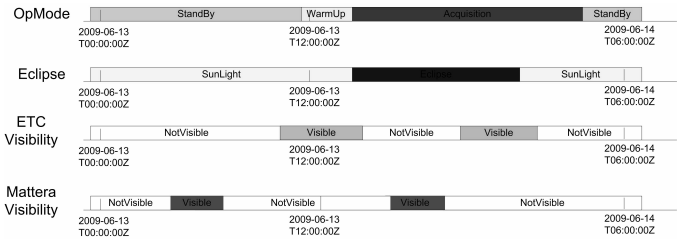

Figure 1: The progression of 4 state variables as timelines.

with a domain indicating the possible values for it. The domain can be an enumeration of string literals or the integer numbers or a subset of integer numbers, etc. The propagation of state variables is done in a deterministic way. This means that SVs only change if some Basic Actions change them. Opposed to that, NCSVs are externally propagated and cannot be changed by any Basic Action. Basic Actions are high level representation of the commands to the satellite that includes not only an indication of which is the command to uplink and its actual parameters, but of which is the effect of executing the command on-board in terms of the effect that we shall expect to see on the SVs.

In the next figure it is shown a hypothetical example of state variables progression. For instance, the NCSV called Eclipse has two values in its domain the string *SunLight* and the string *Eclipse*. In turn, OpMode is a regular SV that is changed only by commands.

Basic Action A Basic Action (BA) is a data set containing the following elements:

1. One *command* from the Basic Command Library.
2. One *reference time point* that is used to represent the moment of its execution.
3. Specifications of its *Execution Conditions*.
4. Specification of its *Effects*.

Being:

- *Basic Command definitions.* Basic Commands are constructed using CONAEs Spacecraft Control Language (SCL), which is a scripting language used between other purposes to define the commands that can be send to the satellite. They can have parameters and several other features;
- *Reference Time Point.* The reference time point represents the moment when the command is executed;
- *Execution Conditions*. An execution condition is specified by means of the value that a state variable of the system should have during a defined time interval.

  When a BA is included in the operations plan, its moment of execution is fixed. As usual, an execution condition is satisfied if the value of the SV in the plan coincides with the value of the condition in all the execution condition time interval.

  In the planning process at the MOC, all execution conditions of a BA are checked to be satisfied. This provide a mechanism for coordination to the Aquarius and SACD Instrument Operations Teams by asking about a particular state of the service platform, that is, to ask for a service. When this happens the Flight Operations Team is responsible to add all necessary BAs to the plan in such a manner to satisfied all instrument teams requests.
- *Effects*. An effect has the same structure as the execution conditions, but with another semantics. It indicates the values the state variables will have because of the execution of a command, and are used to propagate the expected state of the system.

  When a BA is included in the operation plan, the effects of the BA are taken into account and used to modify the value of the state variables.

Lets illustrate the concepts introduced so far with a concrete example. Suppose you have a payload with its own on-board storage capabilities. It is likely that the payload will have a command for beginning to download the data. More precisely, the instrument has a command that sends science data through the bus during the dump duration that is passed as a parameter. Let call the command `Payload_MMDump(dumpDuration)`.

In order to download the science data, synchronization of the `Payload_MMDump` command and the capabilities of the service platform are needed. In fact, a contact with a ground station supporting X-band contact is also needed. In the following figures we see how this synchronization can be specified (and later verified) using the BA representation.

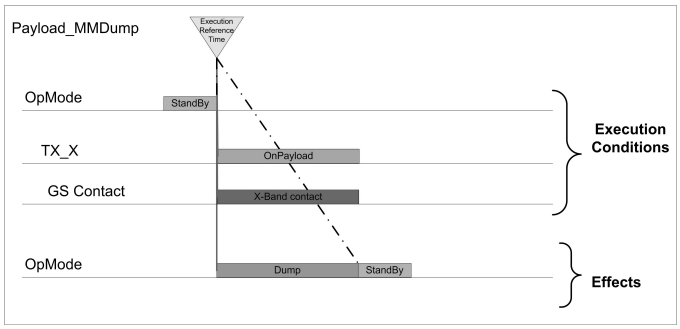

Figure 2: Dump Basic Action Execution Conditions.

Basic Action Component The main constitutive elements of a Basic Action Component (BAC) are Basic Actions. As was explained, a Basic Action corresponds to one command in the satellite basic command library. BACs are intended to be a higher abstraction layer to coordinate the execution of a set of basic actions (i.e. basic commands).

A Basic Action Component is composed by:

- A tuple of Basic Actions.
- Binary time constraints between the basic actions.

Since BA components are used to synchronize the execution of basic actions locally, that in turn translates in the execution of commands; BA components are expected to be strongly constrained (in terms of the time constraints that tell how each BA of the component relates with the others).

Figure 3 shows a Basic Action Component Template (BACT) being built with the GUI developed for this. BACTs provide a way of reusing sequences of commands that implement a given operations. A BACT is a template encapsulation of a parameterized sequence of lower-level BAs such that the parameters translate both to the parameters of the BAs or of the temporal relationship of the sequence of the BAs. Indeed, BACTs can hold not only a parameterized temporal sequences of BAs, but a more complex temporal relationships: a parameterized Simple Temporal Network of BAs. This structure allow for example to represent a partial order of BAs, what –when convenient– can be used for a less commitment approaches to post-pone up to the last time possible to choose between the valid alternatives.

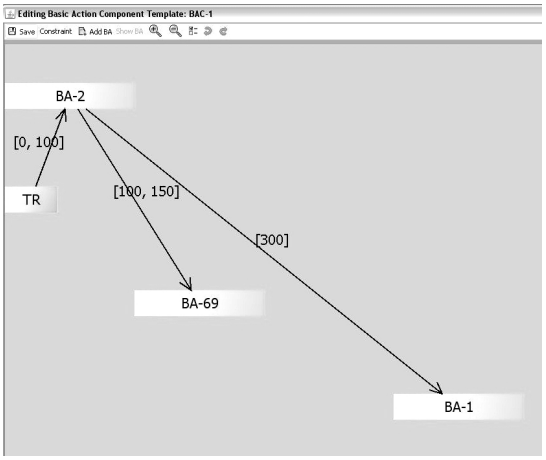

Figure 3: Basic Action Component template being built.

The Basic Action Component concept can be related with EO-1 Spacecraft Command Language module (Chien *et al.* 2003), because it converts a high level data structure into the detailed command sequence to be executed. The BACTs provide a functionality similar to the *decompositions of activities* in the ASPEN system ((Chien *et al.* 2000)), but restricted to only one level of decomposition. Higher level of grouping can be achieved with different construct: the *Action Request* data structure.

An *Action Request* (AR) is a complex structure, allowing many functionalities –including for example the request of the up-link of a software patch or the execution of a real time interactive pass procedure activity–, but its simplest use is to encapsulate many Basic Action Components, i.e. instances of BAC Templates.

To summarize: a BA is the equivalent of an *action* in classical planning but they have a low level implementation in terms of a command and uses SVs and NCSVs instead of predicates when defining its effects and preconditions. A BACT is a Simple Temporal Network of BAs that allows the modular construction of re-usable sequences of commands. An Action Request allows the submission of several BACs (i.e. instantiated BACTs) to be considered for its inclusion in the main plan.

When a Action Request is inserted in the plan (by solving the underlying STP, see (Dechter *et al.* 1991)), the execution conditions and effects of each basic actions are, respectively, controlled and propagated in chronological order. A consistent plan is therefore a plan such that all execution conditions of all BAs are satisfied by the propagated values of the SVs and by those of the NCSVs. The propagated dynamics of the SVs reflect all changes expected in the SVs when the commands are executed.

3 Automatic Orbit-Event-Related Activity Planning

In classical planning ((Fikes and Nilsson 1971)), a *goal* refers to the desired state of the *system* at the end of plan execution, but the subject of how these goals are synthesized, defined or collected is usually not addressed, even if its specification is a central part of the definition of the planning problem. For instance, there might be users that submits requests of images that are first collected, and then, translated into a set of goals or extended goals valid for various planning horizons by systems outside the planning system. The only part included in the specification of the planning problem is the final step, the goal defined for a certain planning horizon. Event if rather complex approaches deal with the so called *extended goals* and *preferences* (e.g. (Baier *et al.* 2009)), the problem of how these extended goals are collected is not treated.

Space missions have science requirements defined by their instruments science teams. In many cases these turn into a description of science objectives and priorities expressed in such a broad way and without considering all operational constraints, that does not allow a direct translation to a concrete plan without some human intervention. This usually implies that without any formal approach an important effort is dedicated to define and adjust the operational interfaces between the science and operation teams.

We believe that modeling science objectives, priorities and constraints, and from that model, to automatically generate the planning problem goals it is an important matter to the success of flexible and efficient space mission planning systems. We will refer to this as the *goal*

generation problem. As we explained in the introduction, one of the two main subjects that we present in this paper is how we solve this problem automatically for the SAC-D/Aquarius mission. Indeed, in this section we show how we deal with the automatic synthesis of the activities to be added in the plan in order to satisfy the science goals.

As mentioned above, goal generation was treated also in (Chien *et al.* 2003) for EO-1 satellite on-board planning system. There, the new targets are defined by processing on-board science data looking for valuable science opportunities. In our case, the automatic goal generation is computed for periodic orbit-related science goals. We implemented a way to link a high level description of orbit related goals with some activities to be added in the operations plan, that are later implemented as Basic Action Components in an Action Request.

To do this, the planning system receives as inputs from CODS (CONAE Orbit Dynamic Services) periodical updates of several files generated from the last TLE (Two Line Element). These include:

1. *GS (Ground Station) visibilities*. These are used to generate X-band downlinks, S-band TT&C, DCS, and PAD (that is the mass memory of instruments) activities. The attributes of a GS visibility include the following: GS ID, start time, stop time, duration, revolution number.
2. *Science target accesses*. These are used for NIRST and HSC cameras activities. For each region of interest described as a polygon, COD Service generates the accesses for a given period and the geometry of the instrument. The attributes of an access include the following: place name, start time, end time, duration, and the four earth points that define a polygon (the region covered by the instrument).

The definition of the activities to be added automatically, taking into account all the information above, follows configurable rules. The rules are used to link each event (GS service and/or target accesses) to the BACTs that implement each activity. The periodicity of the events is a key aspect that allows the transformation. Hence, the rules are applied establishing a period that is the same as the orbital revisit period of seven days.

For example, in Earth Observation satellites it is usual that the Ground Station contacts used for a particular service (e.g. TT&C) follow a fixed pattern that depends on the orbit period. The same is true for some science applications, that require that a particular set of accesses to an area must be acquired periodically.

For example, in the case of SAC-D/Aquarius mission, all ETC ground station visibilities with more than 5 degrees of maximum elevation are used for TT&C services; and three Matera ground station visibilities are used for X-Band downlink services. Specifically, if the passes in a given orbit period are enumerated, the indexes of the passes used for these services are fixed. For example, of all 28 ETC passes in a week, all except pass 2 and 11 are used for at least one of the five types of possible X-Band downlinks combinations. The number of downlink combinations comes from choosing some subset of the instruments which data is downloaded multiplexing, but not all combinations are legal. What passes to use for each instrument is predefined by the X-Band downlink budget that was distributed between all instruments.

Another important aspect is that the system shall allow the complete reconfiguration of these goals, that can change drastically along the mission life-time. In other words: the rules to generate these operation goals need to be reconfigurable. After considering several use-cases, we defined the rules called *Activity Instantiation Rules*, that give the flexibility needed to automate the process. These rules are the key of the automatic generation of activities, and are explained in detail below.

The instantiation rules key fiels are the following:

Event ID. This is the ID of the event that implies the generation of a new activity. For example, a visibility with a particular GS, or an access to a given ROI (Region of Interest).

Activity Type. The type of the activity that must be generated. These are activities of a high level of abstraction, that are later implemented in terms of BACTs (the lower-level detailed representation the planning system handles for plan verification and command uplink compilation).

Time Series. These include a time window for which the activity must be generated and the periodicity used to enumerate the events. That is, inside each period defined by the time series the events are enumerated chronologically.

Star Time Binding. This is a data structure that specifies how the attributes of a given event are used to define the start time of an activity. Any date field (e.g. start time, end time) of an event can be bound to the start time of the activity. Furthermore, a delay can be defined using any mathematical formula on the numeric attributes of the event (float, int, etc). This flexibility was added because some activities needed to be allocated at the middle point of a pass, and another activities uses the end time of the event as the time point reference.

Parameter Bindings. This is a list of bindings that allow to specify how the attributes of the event are bound to the parameters of the activity. A lot of flexibility is allowed, including fixing values or values proportional to the duration of the event. For example, this is used to transfer the duration of an access to a target to a parameter of a BACT, that in turn connects this parameter with the duration parameter of the command to execute an instrument acquisition.

Selected Occurrences. These are the indexes that indicate what occurrences of the events are used to generate activities within one period.

Minimum Access Time. This is a filter that is applied to the duration of the events before the enumeration is done. It allows to leave out spurious short events that can emerge due to small variation in the orbit. It gives robustness to the event enumeration.

With the orbital data and the *Activity Instantiation Rules*, the system is able to generate a list of activities. This list is later translated to *Action Requests* (see previous section) containing various Basic Action Components instances in agreement with these rules. This is the format that the core planning system ingests to perform the final validation of the plan by propagating state variables and control if all constraints holds, and later, to generate the pass scripts with the command that will be upload.

The translation is done by another configurable feature that links each Activity to a BACT. That is, orbit events are linked to activities, and in turn, activities are linked to BACT implementations. This is done in a two steps process to allow changing one relation without changing the other. For example, the Flight Operations Team can change the sequence of commands that implements a given activity, and the idea was to allow changing that without having to change the rules that generate the activities. Hence, in this way, the specification of some periodic activities are linked with the BACT concept, which is the data structure that connects the high level representation of the plan with the low-level implementation in terms of sequences of Basic Actions (i.e. sequences of commands).

All these steps are automatic –just a few mouse clicks to start the process are required–, and the generated plan is validated by checking resource usage of each instrument, i.e. we check if they are in agreement with the assigned budgets. If a constraint is violated, an activity is left out of the plan. This is also an easy task because each activity inherits a priority level from the orbit event parent. The activity with the lower priority level is erased.

This plan, implemented in terms of BACs is then validated with the BA models, that gives a causal validation at the lowest possible level of abstraction (i.e. the command level).

Figure 4 shows a data flow diagram presenting this function.

The system is currently allowing the automatic insertion of a few hundreds of activities per week, that were managed manually or semi manually before.

An example of these generated activities is the real-time HSC camera acquisitions. The acquisitions are called *real-time* because the instrument downloads the image while is being taken, and it is not stored. For this reason, the periodic real time acquisitions are associated to some specific previously agreed ETC passes (specifically,

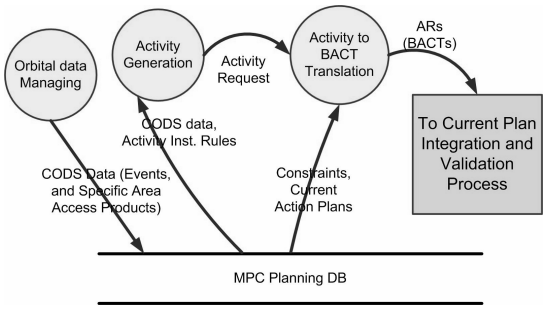

Figure 4: Data flow diagram of the activity generation function.

descendant passes 1, 5, 9, 14, 23, 27). The orbit event (i.e. the ground station passese), contains the following data:

```
<Event>
   <eventID>ETC</eventID>
   <start>2012/04/03T11:21:02.0</start>
   <stop>2012/04/03T11:28:02.0</stop>
   <duration>0:07:00</duration>
   ...
</Event>
```

For descending passes, because of the sensibility of the camera, it is better to do the acquisition as close to the LOS of the pass as possible. This means that the reference point to start the acquisition is the *end* of the orbit event. For pass elevation considerations, the acquisition must start 702 seconds before the LOS.

The command for an acquisition has a parameter named *PassDuration* that is bound to the fixed value of 600 seconds. Because of thermal matters, the camera can only acquire ten minutes.

The following is the configuration file that specify all that constraints.

```
<codsEventActivityInstance>
   <aic:activityType>HSC_RT_Acquisition
            </aic:activityType>
   <timeOfInterest>
      <ts:timeSeries>
         <ts:surveyPeriod>
            <ts:min>2012-11-05T00:00:00
                     </ts:min>
            <ts:max>2014-12-31T23:59:59
                     </ts:max>
         </ts:surveyPeriod>
         <ts:occurrences>2</ts:occurrences>
         <ts:periodicityRange>
            <ts:min>7.0</ts:min>
            <ts:max>7.0</ts:max>
         </ts:periodicityRange>
      </ts:timeSeries>
   </timeOfInterest>
   <startTimeBinding>
      <aic:binding>stop</aic:binding>
      <aic:delay>-702</aic:delay>
   </startTimeBinding>
```

```
   <aic:parameterBindings>
      <aic:parameter>
         <aic:activityParameter>PassDuration
               </aic:activityParameter>
         <aic:activityParameterValue>600
               </aic:activityParameterValue>
      </aic:parameter>
   </aic:parameterBindings>
   <aic:filters>
      <aic:minimumAccessTime>426
               </aic:minimumAccessTime>
      <aic:selectedOccurrences>1,5,9,14,23,27
               </aic:selectedOccurrences>
  </aic:filters>
...
</codsEventActivityInstance>
```

4 Plan Execution Validation

This second functionality of the new planning system that we are presenting here can be summarized as follows:

> The system shall allow the traceability of the correct execution of telecommands. More specifically, the system shall have the capability of processing telemetry in order to verify the successful *execution* of the commands in the plan (whenever there is telemetry that allows the verification without ambiguities).

Plan execution validation is implemented in many planning system that are used in practice like in (Chien *et al.* 2000), (Jónsson *et al.* 2000) and (Chien *et al.* 2003). Our approach does not introduce a brand new concept, but we consider important to describe the design details of our approach with the objective of allowing other people to contrast with theirs, compare advantages and disadvantages, and through this process helping with the development of better systems. Even if at a first approach we might consider that this is not part of the core of planning, this is not the case when we understood that the modeling done in planning provides most key things for this validation. Both things are critically coupled. The planning system at the MOC usually is the only system that has the knowledge of what is *expected* to happen on board. This is the only system capable of forecasting in detail the expected behavior of several key parameters in tied connection with the commanding plan that it is synthesized.

Furthermore, to be useful, the feedback about the plan execution that is contained in the stored telemetry must be later impacted in the plan. This means that some kind of translation from the satellite telemetry variables to the planning SVs is needed. For example, when something went differently from what was planned, it is needed to re-plan taking into account what actually happened: we need to correct some SV which values were propagated at planning time, replacing its expected values with the observed ones.

The approach we used for implementing the plan execution validation new features mirrors that we used for commanding. We associate low level satellite telemetry variables values with the values of the SVs and compare them appropriately.

At the MOC, a specialized scripting language called Telemetry Specification Language (TSL) is used to define telemetry. The MOC tools use these definitions to decode the values of the telemetry variables. This language includes several libraries with specific functions for this task. It includes all mathematic functions of any programming language together with functions for easy bit/byte low-level manipulation and bit/byte extraction from the telemetry frames.

First, we identify already existing telemetry variables whose values corresponds with higher level SVs. With the remaining set of SVs whose values that do not match exactly with lower level telemetry variables definitions, we defined derived telemetry variables that actually match its values exactly.

The final result is a subset of all the telemetry variables whose values precisely correspond with each SV value at the planning system. Notice that the domain of these telemetry variables corresponds exactly with the domain of the corresponding state variables.

For the comparison, it is important to consider that all telemetry variables have an intrinsic latency that is not necessary the same (some commands effects can be observed in the telemetry after a few seconds and for others it might be necessary to wait hours). We use some configuration parameters for each SV to register this natural latency and only compare when adequate.

When the telemetry is processed, a telemetry product with the decoded values is created. This is an xml file containing all records of the TSL decoded telemetry within a time frame. This product is later processed by the new module of the planning system called *Uplink and Execution Analyzer*. If there is a difference between the state variable propagated at planning time and the report based on telemetry, the plan is updated.

The configuration file that links a telemetry variable with a state variable includes the following fields:

- *SV id*. The id of the state variable that is used for the comparison.
- *Tlmy Var id*. The id of the telemetry variable to be compared with the SV values.
- *Accepted Latency*. Two integer fields (named $after$ and $before$) that indicate the pre and post SV change accepted margin (in seconds). If the SV change its value from val_1 to val_2, at instant t_0, the system looks for the same value change in the interval $[t_0 - after, t_0 - before]$. This gives the flexibility needed because of the latency in the recorded telemetry, and the small difference that could

also be introduced in the exact execution of the command.

- *Accepted Margin*. These are two numeric fields (named $upper$ and $lower$). These are meant only for SV/TlmyVar with numeric domains (int, float, etc). They add some margin in the changes in the values to be searched. The exact meaning is that, if a numeric SV change its value from val_1 to val_2, at instant t_0, the change to be search in the actual telemetry is a change from a value $\hat{val}_1 \in [val_1 - lower, val_1 + upper]$ to another value $\hat{val}_2 \in [val_2 - lower, val_2 + upper]$. This margins are needed because power consumptions or temperatures cannot be predicted with extreme accuracy at planning time. Notice that the margins that are considered are uniform for the SV/TlmyVar pair, and does not depend on the considered value of the variable. This is a simplification that is acceptable for our case and that other applications might consider different.

All the above parameters can be easily identify by studying the variations in the actual telemetry received from the satellite.

Remember that at *checking time*, the SVs time-lines include the value changes implied by all basic action effects, and fulfill all basic action execution conditions. This means that any unexpected change in the comparison would mean an unexpected behavior in the execution of the commands that cannot bee detected by the usual control of off-limit alarms.

5 Future work

For future CONAE's SAOCOM L-Band SAR mission, its MOC will be receiving from users a lot of requests in terms to areas of interest to be covered by the instrument data takes. The decomposition of the areas of interest in several acquisitions can be done as presented in this paper, but the decomposition will lead to an over-subscribed plan. We envisage to add a tool for the automatic selection of the activities that produces an optimal plan considering all users preferences and their priorities and quotas. Currently, we are investigating the application of various techniques to model this problem.

6 Conclusion

In this paper we described the design and implementation of two new functionalities added to the SAC-D/Aquarius mission planning system, highlighting several details that contribute to the system robustness, and how they are used in our mission. These functionalities are:

1. The automatic synthesis of the satellite activity plan from orbit events, including the generation of the instruments acquisition plan needed to acquire the science targets. We also explained how these plan is further translated into a low level plan of commands to be uploaded. This approach can be applied in the automatic generation of orbit related activities (such as down-links) for any low-orbit satellite mission with periodic orbit (virtually all earth observation science satellite missions). This not only reduces the work load, but also reduces manual human errors, contributing to the robustness of the mission operations.
2. The validation of the correct execution of the satellite activity plan, by comparing the values in the stored telemetry with the expected values computed at planning time.

The first is already operational and the second in the final validation and deployment phase.

References

Jorge A. Baier, Fahiem Bacchus, and Sheila A. McIlraith. A heuristic search approach to planning with temporally extended preferences. *Artif. Intell.*, 173(5-6):593–618, 2009.

D.G. Boden and W.J. Larson, editors. *Cost-Effective Space Mission Operations*. McGraw-Hill,Inc, 1996. Space Technology Series.

S. Chien, G. Rabideau, R. Knight, R. Sherwood, B. Engelhardt, D. Mutz, T. Estlin, B.Smith, F. Fisher, T. Barret, G. Stebbins, and D. Tran. ASPEN - Automated planning and scheduling for space missions operations. In *International Conference on Space Operations (SpaceOps 2000)*, Toulouse, France, June 2000.

S. Chien, R. Sherwood, D. Tran, R. Castano, B. Cichy, A. Davies, G. Rabideau, N. Tang, M. Burl, D. Mandl, S. Frye, J. Hengemihle, J. D'Agostino, R. Bote, B. Trout, S. Shulman, S. Ungar, J. Van Gaasbeck, D. Boyer, M. Griffin, H. Burke, R. Greeley, T. Doggett, K. Williams, V. Baker, and J. Dohm. Autonomous science on the EO-1 mission. In *International Symposium on Artificial Intelligence Robotics and Automation in Space (i-SAIRAS)*, 2003.

R. Dechter, I. Meiri, and J. Pearl. Temporal Constraint Networks. *Artificial Intelligence Journal*, 49:61–95, 1991.

F. Dvorák and R. Barták. AI planning with time and resource constraints. In *Proceedings of the Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems (COPLAS 2010)*, pages 5–13, June 2010.

R.E. Fikes and N.J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence Journal*, 2(3-4):189–208, 1971.

Ari K. Jónsson, Paul H. Morris, Nicola Muscettola, Kanna Rajan, and Benjamin D. Smith. Planning in interplanetary space: Theory and practice. In Steve Chien, Subbarao Kambhampati, and Craig A. Knoblock, editors, *AIPS*, pages 177–186. AAAI, 2000.

W.J. Larson and J.R. Wertz, editors. *Space Mission Analysis and Design*. Microcosm Press and Kluwer Academic Publishers, 1999. Third Edition.

D.J. Mandl, R.A. Sohlberg, C.O. Justice, S.G. Ungar, T.J. Ames, S.W. Frye, S. Chien, D.Q. Tran, P.G. Cappelaere, D.V. Sullivan, and V.G. Ambrosia. A space-based sensor web for disaster management. In *In Geoscience and Remote Sensing Symposium. IGARSS 2008. IEEE International*, volume 5, pages V.294–V.297, July 2008.

E. Romero and M. Oglietti. Cooperative space mission operation planning by extended preferences. In *6th International Workshop on Planning and Scheduling for Space (IWPSS 2009), (poster presentation)*, July 2009.