Cooperative Space Mission Operation Planning by Extended Preferences.

Eduardo Romero, Marcelo Oglietti
CONAE - Argentine National Space Agency
Paseo Colon 751, (1063) Buenos Aires
Argentina
eromero@conae.gov.ar
marcelo.oglietti@conae.gov.ar

Abstract. Joint space missions that include the participation of several space agencies are very usual nowadays. A particular example of these are some low earth-orbit satellite missions. In these missions, the level of participation and the scope of responsibilities of the space agencies involved can be very heterogenic.In this paper we present the usual problems that arise in the ground operations of joint low earth orbit satellite missions. We also show how a few modifications in the architecture of the Mission Control Center and a common planning modeling can be used to overcome this problems. By augmenting the concept of preference, we obtain a flexible and distributed mission operation planning. We briefly describe an application of these ideas to the ground operations concepts of a joint space mission close to be launch: the AQUARIUS/SAC-D sun-synchronous low orbit satellite mission that will have eight different instruments onboard.

1 Introduction

During the last decades, joint space missions with the participation of several space agencies and companies have been growing in number and size. This has been pushed by cooperating governments that seek the same global objectives such as environmental or deep-space studies. Joint missions are seen as a great opportunity for cost reduction, technical knowledge interchange, and international cooperation encourage, being the International Space Station the emblematic example of international space cooperation.

Nowadays, important examples of this kind of missions are the Low Earth Orbit (LEO) satellite missions. In a LEO satellite mission, it is usual to find several science instruments onboard the spacecraft that are owned by different space agencies or companies. These instruments or payloads compete in the use of the onboard resources such as power, tele-command storage capabilities, and science-data storage capability; and ground resources such as communication bandwidth for tele-command uplink and science data download. For this reason, the operations of a joint LEO satellite mission need to be coordinated between the science instrument teams and the Mission Operation Center (MOC).

The usual solution implemented for this problem is to simplify as much as possible the interfaces between the spacecraft and the payload and to extremely simplify the nominal operations of the payload. In these conditions, the operations are completely transferred to a unique Mission Operation Control team that constructs the plan for the whole spacecraft considering a few simple guidelines on the use of the payload. The simplifications are implemented by design and usually imply that the potential of the instrument is not fully deployed.

For these reason, among others explained in the next section, the use of a distributed planning framework would represent a substantial enhancement in the operations of a joint LEO satellite mission.

On the AI Planning & Scheduling side, most of the efforts on distributed planning have been focused on multi-agent domains (see for example (Brenner 2003; Brafman and Domshlak 2008)). In that context, distributed planning embraces the generation of a plan in a domain were several agents can execute actions and the actions must be coordinated in the seek of a unique plan; and planning algorithms that exploit that characteristics are researched.

In the case of cooperative space mission operations, the situation is different. Similarly to multi-agent domains, we have several teams generating incomplete partial plans that have to be integrated in a main unique plan. But we also have the team in charge of the service platform, that has to integrate all partial plans considering constraints and purposes that are out of the scope or even unknown to the payloads. Making an analogy with agents, we have several agents (the payloads), and we need a scheme were each agent can propose (and require) a partial incomplete plan that wants to integrate in the main plan of the system. The agent has to be able to require services to perform its partial plan that doesn't know how are actually implemented.

In this paper we explain some architectural concepts for ground operations and a framework that allows the coordination of the operations in a distributed planning scheme. We also show how this concepts are applied in the ground segment of AQUARIUS/SAC-D mission, a CONAE, NASA, ASI, and CNES joint LEO satellite mission.

2 The Need of Pro-Distributed Planning Interfaces

As was said in the introduction, we will focus on the Low Earth Orbit (LEO) science satellite missions.

The Spacecraft. A LEO science satellite is usually designed following a modular architecture with several subsystems. This modularity is fundamental for the separated manufacture of the components by different parties, and for making possible the later integration of all the components.

The subsystems that are in charge of maintaining the satellite's health and of providing the basic services are grouped to compose the so called *Service Platform* (SP). The SP includes a power subsystem, an attitude subsystem, a mass memory subsystem, and so on. The SP provides a set of clear interfaces to integrate into the satellite the instruments (also called payloads).

The SP has to maintain the satellite in orbit, has to provide services to each payload, and has to record its internal state (the telemetry) periodically for future analysis.

The architecture is extremely modular and, as long as the SP concerns, the instruments are just black-boxes requiring power, attitude modes, down-link transmitter use, and in some cases, data storage and *Time Tagged Command* (TTCmd) storage and administration. If available for the mission, TTCmd storage and administration and telemetry recording is done by a very important subsystem, usually called *Command, Control & Data Handling* (Boden and Larson 1996) or simply *Command & Data Handling* (CDH).

Even if command storage is delegated to the SP, in order to command the instruments, there is no need of knowing exactly what the instrument does, as long as the requirements of the instrument are satisfied in terms of power, thermal, orbit, etc.; and as long the instrument behavior is among the valid limits in terms of power consumption, heat generation, data-generation rate, etc.

Cooperative LEO satellites missions exploit this onboard modularity to make possible the integration of the SP with instruments that are designed and constructed by different partners.

The Ground Segment. A space mission has a Flight Segment (FS) and a Ground Segment (GS). GS architecture may vary substantially from mission to mission, but there are some established space engineering methodologies that are widely spread described in many books, e.g. (Larson and Wertz 1999; Boden and Larson 1996), and underlined in some international standardization efforts like the European Cooperation for Space Standardization (ECSS) or the Consultative Committee for Space Data Systems (CCSDS).

In the GS, there is a system in charge of constructing the whole plan of operations for the spacecraft and of executing this plan. We refer to this system as *Mission Operation Center* (MOC), although it can usually be presented with

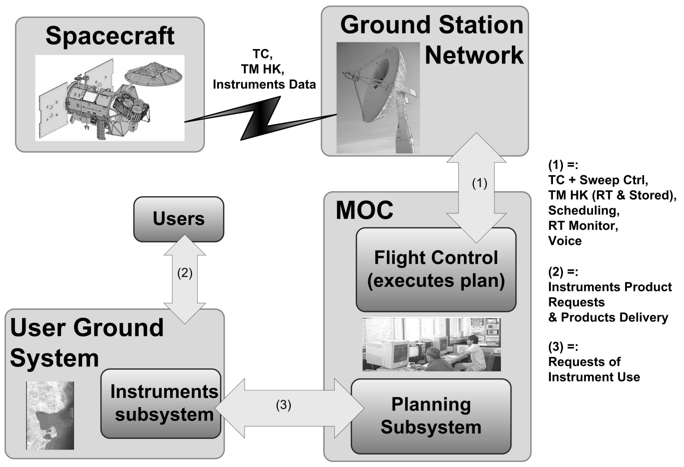


Figure 1: Ground Segment General Architecture

other similar names like, for instance Mission Control Center. A MOC consists in the people, hardware, software and infrastructure that is needed to Monitor and Control (M&C) a satellite from earth; leaving outside the Ground Networks that give support to the mission (i.e., antennas and RF equipment and personnel is not part of the MOC). Among other tasks, the MOC is in charge of negotiating ground contacts with the ground networks that give support to the mission.

The MOC receives as an input for plan generation the science requirement from payload teams. Payload teams are usually closely related with the system of the GS in charge of providing the data to the final users (we refer to it as *User Ground System* (UGS)). The science requirements can be very general, such as a few guidelines for the constant use of the instrument; or very precise, such as a list of requested data acquisitions for particular moments. But usually, the requirements are defined with a high level of abstraction, avoiding the complicated details of the implementation of that requirements in the operations of the mission. Examples of these details are the specific set of commands that have to be sent to the instrument or the operational constraints (eclipses, attitude modes, ground station visibility, etc). We placed Instrument teams as functionally located at UGS, because they are not common users, but contribute to the construction of the requirements for the plan generation at the MOC. They have a detailed knowledge of the instrument, but usually its lack of detailed knowledge on the SP does not allow them to intervene directly on the details of the spacecraft operation plans.

After spacecraft commissioning the operations are completely transferred to GS. The MOC receives a specific set of procedures from each instrument team to follow for constructing the spacecraft operation plans from the science requests. This results in a very rigid scheme were not all the possibilities of the instruments are exploited and the operations turn to be more rigid than expected.

There are also other reasons why this simplified solution sometimes does not fit with the requirements of the mission.

1. Each payload team wants to exploit its payload use at maximum.

   The delegation of the operations of the whole spacecraft to only one team, without the intervention of the instrument teams, usually turns out in a conservative use of the payloads. This is because the responsibility of maintaining and extending the health of the spacecraft bias the whole operations to the detriment of efficiency. Hence, separating the competing responsibilities of planning for taking care of the State Of Health (SOH) of the spacecraft and planning for using a payload, increase the efficiency of the operations.

2. Operations sometimes require a detailed knowledge of the payload that only the owner has.

   Specific data acquisition for calibration tasks, or onboard-software patch uploads are just two examples of critical operations which know-how is usually on the instrument owners side.

3. The need of delimiting responsibilities.

   In the case of a very delicate or complex to command instrument, it is sometimes desirable to refer the complete operation of the payload to its owner. Especially if a wrong operation of the instrument can lead to a problem with the instrument state of health.

By transferring all the details of operations plan generation to the MOC, the modularity that allows the integration of the whole spacecraft does not has a counterpart on the GS architecture.

In order to overcome this issue, we find out that the key point lies in the interface between the MOC and the UGS, with the instrument teams, i.e., the (3) interface in Figure 1. In other words, depending on what is intended as *Request of Instrument Use*, the planning of the operations can be more or less pro-distributed.

By letting the instrument teams to participate in the construction of the plan at a very detailed level, we can overcome all the above mentioned problems. That is, we need to *distribute* the planning process. But, as was told before, the fact that instrument teams do not have the knowledge nor the responsibility on the way the SP has to be operated, the construction of the plan needs to be modular too.

For example, suppose the satellite has a camera onboard and a real-time image is required (i.e., an image that is downloaded during the acquisition in real-time over Ground Station visibility). The usual practice will be to require just a real-time image to the MOC specifying the beginning and ending times and the mode of the acquisition, and let the MOC to manage the details of the acquisition implementation. This is done in this way because a real-time acquisition requires more than just turning on and off a camera, it requires commands to turn on and turn off the transmitter that are intrinsic to SP operations. Even more, and complex, SP commands could be needed if the imaging requires some attitude manoeuver.

Besides, there is usually a lot of flexibility in some payload operations that is not easy to directly translate to the low-level detailed plan. For example, a dump instrument's science data could be wanted to occur within a time window rather than on some specific contact. If payload teams specify a dump for a particular contact, since they do not intervene in ground station coverage negotiation, the plan could easily turn out to be unfeasible. In other words, the flexibility that is present in the high-level description of the operations is fundamental to achieving a reasonable plan of operations at the MOC.

Hence, if we want to allow the specification of more details in the requests for instrument use, we need to transfer the portion of the plan of the instruments to the instrument teams at the UGS. But in order to do that we need a framework where the following features are available.

1. A clear, flexible and human-readable specification of the particular set of commands that the instrument can execute. This means the existence of a common agreement on the details of the instrument operations.
2. Specification of time constraints about the execution of the commands.
3. Specification of needed conditions on the SP for the execution of the commands.
4. A way of re-use sets of commands that are usually used for the same tasks.

In the next section, we explain how we have attacked this problem by redefining the interface and implementing a distributed planning scheme with extended preferences.

3 More than Preferences

In classical planning (Fikes and Nilsson 1971), a *goal* refers to the desired state of the *system* at the end of plan execution. Goal verification is the final objective of plan construction. The concept *Extended goal* refers to the specification of some mandatory properties of the states that the system has to go through (or avoid) during the application of the constructed plan (Bacchus and Kabanza 1998). If we see the extended goals as hard constraints on the resulting plan, planning with extended goals is closely related with planning with *preferences* (Baier *et al.* 2009), where the preferences can be viewed as soft constraints over the resulting plan. This means that preferences are *desired* properties on the sequence of states resulting of plan plan execution.

In order to model distribution in the planning process, the execution of some specific actions (the actions the payload wants to execute) have to be added as an input for the construction of the main plan. This implies the need of extending the concept of planning with preferences to include not

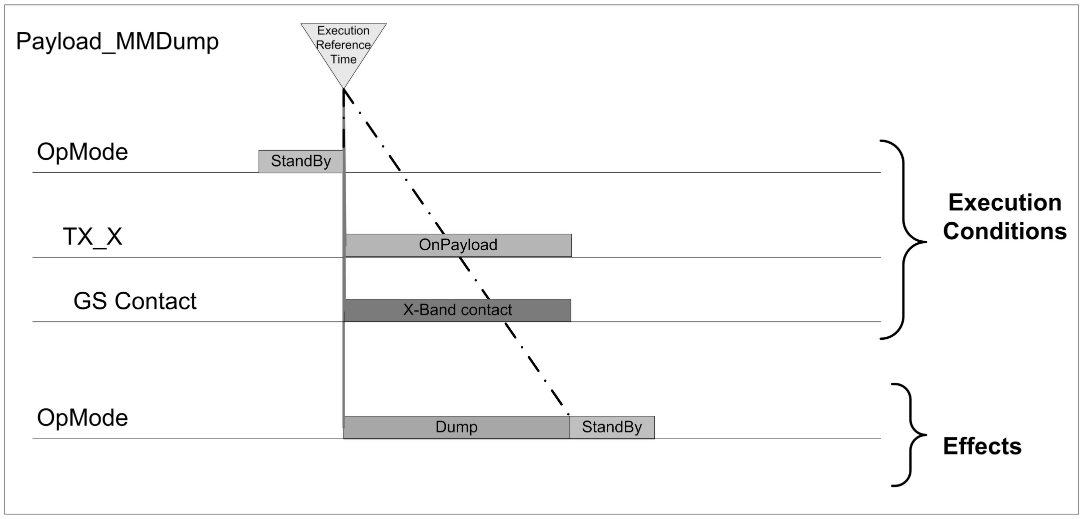


Figure 2: Basic action for a payload mass-memory data dump.

only specifications of properties of the intermediate states, but specification of which actions are wanted (or desired) to be executed.

For this reason, we use an extended notion of extended goal, that we call *Action Request* (AR). An AR is, broadly speaking, an incomplete, not entirely sound or grounded, partial plan. ARs are used by the payload teams to specify what they want to include in the operations plan.

In order to do so, we use a modular description of the planning domain. Each payload and subsystem state is modeled as the aggregation of *state variables* (an usual practice when planning with time and resources (Ghallab *et al*. 2004; Cesta *et al*. 2003)). There is also a set of *non-controllable state variables* that are propagated externally and are used to represent external uncontrollable but predictive events, such as eclipses, ground station visibility, ground station arrangement contacts, etc.

The state variables have a specific relation with the telemetry variables that allows a post-execution analysis of the spacecraft operation plans. Each payload and subsystem has a set of *basic actions* that have a low-level implementation (the tele-commands) and durative *execution conditions* and *effects*. The execution conditions and the effects are expressed in terms of the state variables. Both, execution conditions and effects, are defined by a state variable identifier, a value from the state variable domain, and a duration. As usual, execution conditions have to be verified for the basic action to be executable; and the effects are assignments to the state variables after the execution of the action.

There are also *extended conditions*, that are similar to the execution conditions of basic actions, but are not attached to any basic action and can express the desired value of any state variable of the system. An important feature is that extended conditions can also specify values for non-controllable state variables. These are used for a payload to ask for environmental conditions for the execution of an action.

These are the basic bricks that the payload teams use to construct Action Requests. An Action Request is defined as a set of basic actions and extended conditions, together with *temporal constraints* encoded as STP networks (Dechter *et al*. 1991) that give structure to these elements.

An AR is *included* in a plan by instantiating the execution times of each basic action in a way that satisfies all temporal constraints of the AR . The MOC planning task is to construct a main spacecraft operation plan that maximize an objective function on the attended Action Requests (that is, roughly speaking, including as much as possible ARs from the ARs that the instrument teams send, but with the use of weight on the ARs). For this to be possible, several basic actions belonging to the Service Platform's library have to be added too, to guarantee the conditions of execution of the AR.

In the next section, we briefly review an application of this scheme, the planning of Aquiarius/SAC-D operations.

4 The AQUARIUS/SAC-D Mission

The Aquarius/SAC-D Mission is a cooperative mission. The main partners are CONAE (Comisin Nacional de Actividades Espaciales the Argentine National Space Agency and NASA (National Aeronautics and Space Administration, the United States National Space Agency). ASI (Agenzia Spaziale Italiana, the Italian National Space Agency); and CNES (Centre National d'Etudes Spatiales, the French National Space Agency) also participate in the mission with two science instruments.

Aquarius/SAC-D Mission consists of a LEO satellite flying at 657 km on a 98 deg Sun-synchronous polar orbit.

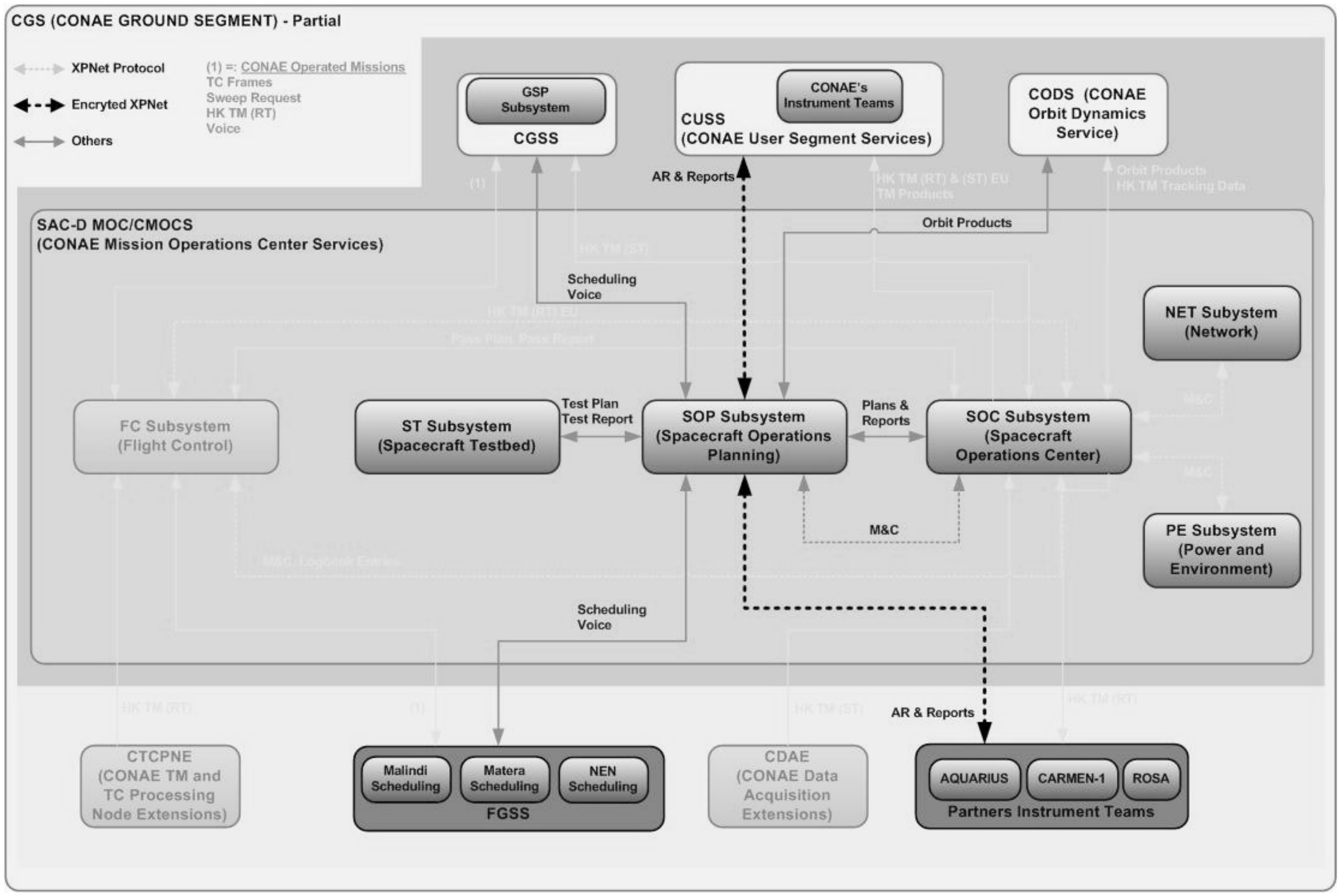


Figure 3: SAC-D GS Architectural Design.

The satellite has eight instruments onboard. The satellite is scheduled to be launched on May of 2010.

The primary science objectives of the mission are to contribute to the understanding of the whole Earth system and the effects of natural and human-induced changes on the global environment. Specifically, the mission will conduct observations of the Earth in order to obtain new information on climate by measuring sea surface salinity, and will resolve missing physical processes that link the water cycle, the climate, and the ocean. SAC-D must also identify hot spots on the ground surface to allow the mapping of fire risk, and perform measurements of soil humidity to prevent floods.

The Aquarius/SAC-D 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).

MWR, NIRST, HSC, DCS and TDP are CONAE's instruments. CARMEN-1 is provided by CNES and ROSA is provided by ASI. Aquarius instrument is provided by NASA.

These instruments have, among many others, 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, 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, etc.

The spacecraft is composed by the eight instruments and the Service Platform (SP). The SP construction is in charge of CONAE and the operations of the spacecraft will be carried out by CONAE Ground Segment, but the plan of operations of all the instruments from other space agencies will be generated by each corresponding instrument teams. Instrument teams has indeed its own ground system that uses the *Action Requests and Reports* interface to dialogue with the MOC to ask specific operations.

The architecture of the ground segment is shown in Figure 3. As can be noticed it follows the already explained concepts. In the diagram, the MOC subsystem is shown at subsystem level, and the Spacecraft Operations Planning (SOP) subsystem and its interfaces are highlighted. CONAE's instrument teams are functionally located at CONAE User Segment Services (CUSS) and Aquarius, CARMEN-1 and ROSA teams are functionally located at the Foreign User Segment Services (FUSS). For both systems the interface is through Action Request and Reports.

Although it is out of the scope of this paper, SOP subsystem deserves a few words because of its importance in the planning process. Among many units that perform all the needed tasks for planning (ground station negotiation, Action Request administration, orbital data collection, pass SCL script compilation form the current plan, etc) there is a GUI called Human Planner Interface that allows the human planner to manually construct the plan in an iterative refinement process, writing the changes in the Operations Plan Database. Basic Action conditions of execution, Extended Conditions and constraints (such as power consumption) are then propagated in order to assure they are properly met.

In the next lines we show how each of the features which need was identified in section 2, is covered in this implementation.

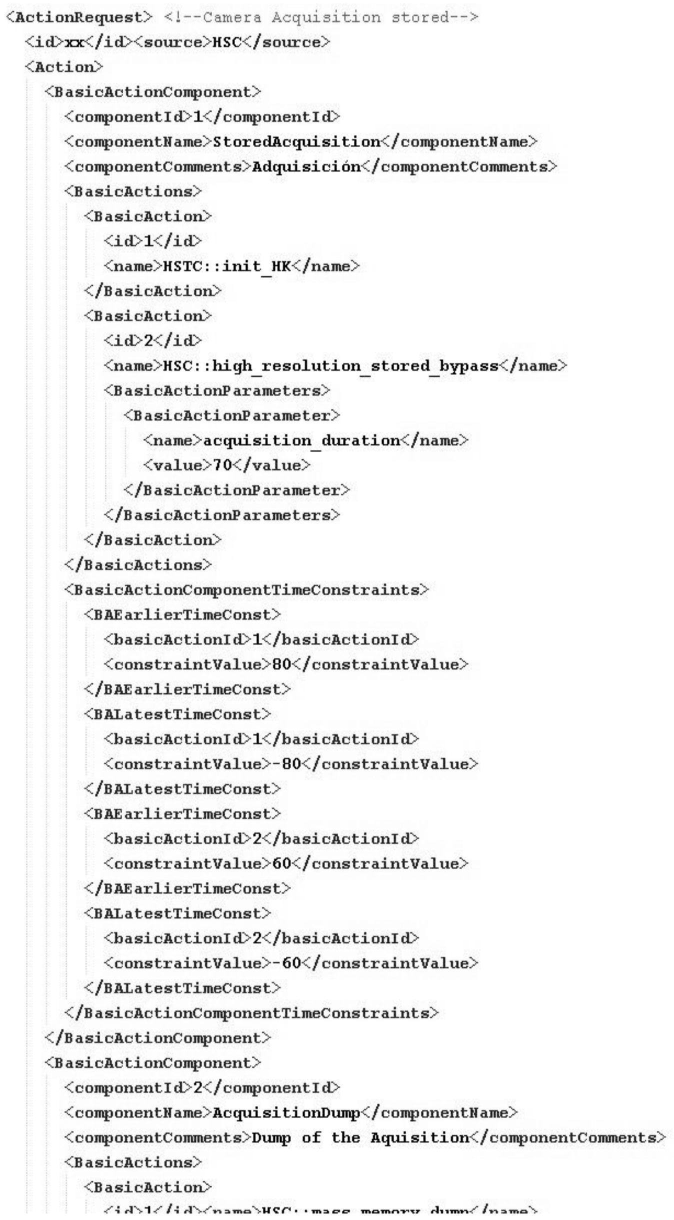
1. A clear, flexible and human-readable specification of the particular set of commands that the instrument can execute.

   SAC-D MOC Flight Control subsystem (FC) provides a tool for soft-encoding all spacecraft tele-commands. This is a powerful tool to easily translate binary tele-command to human-readable self-explanatory commands with parameters. This commands are saved in configuration control at the MOC and they are organized by instrument and subsystem. This tools represent also a great help for the Integration and Test process because its flexibility.

   Basic Actions are also kept under configuration control and are defined by using the tele-commands as the low level implementation of them, but with the addition of execution conditions and effects. This relation between commands and basic actions provides the link between the description of the plan that SOP subsystem and Instrument Teams manage with the low level details of activities of operations.

2. Specification constraints between commands' execution times.

   The AR messages allow several types of time constraints that are more or less expressive and flexible depending on the level of abstraction of the constraint. Basically, it allows the coordination of sets of basic action in a very restricted simple temporal network. These sets of actions



```
<ActionRequest> <!--Camera Acquisition stored-->
  <id>xx</id><source>HSC</source>
  <Action>
    <BasicActionComponent>
      <componentId>1</componentId>
      <componentName>StoredAcquisition</componentName>
      <componentComments>Adquisición</componentComments>
      <BasicActions>
        <BasicAction>
          <id>1</id>
          <name>HSTC::init_HK</name>
        </BasicAction>
        <BasicAction>
          <id>2</id>
          <name>HSC::high_resolution_stored_bypass</name>
          <BasicActionParameters>
            <BasicActionParameter>
              <name>acquisition_duration</name>
              <value>70</value>
            </BasicActionParameter>
          </BasicActionParameters>
        </BasicAction>
      </BasicActions>
      <BasicActionComponentTimeConstraints>
        <BAEarlierTimeConst>
          <basicActionId>1</basicActionId>
          <constraintValue>80</constraintValue>
        </BAEarlierTimeConst>
        <BALatestTimeConst>
          <basicActionId>1</basicActionId>
          <constraintValue>-80</constraintValue>
        </BALatestTimeConst>
        <BAEarlierTimeConst>
          <basicActionId>2</basicActionId>
          <constraintValue>60</constraintValue>
        </BAEarlierTimeConst>
        <BALatestTimeConst>
          <basicActionId>2</basicActionId>
          <constraintValue>-60</constraintValue>
        </BALatestTimeConst>
      </BasicActionComponentTimeConstraints>
    </BasicActionComponent>
    <BasicActionComponent>
      <componentId>2</componentId>
      <componentName>AcquisitionDump</componentName>
      <componentComments>Dump of the Aquisition</componentComments>
      <BasicActions>
        <BasicAction>
          <id>1</id><name>HSC::mass_memory_dump</name>
```

Figure 4: A partial view of an AR for HSC camera Stored Acquisition.

with the restrictions are called *Basic Action* (BA) Components. Specifically, a BA component comprises the following:

- A tuple of Basic Actions, $(\beta_1, \ldots, \beta_n)$, all of them from the instrument's basic action library.
- Binary time constraints between the basic actions of the following form

  $$c_{ji} \leqslant t(\beta_i) - t(\beta_j) \leqslant c_{ij},$$

  where $t(\beta_i)$ is the moment of execution of β_i; and $c_{ij}, c_{ji} \in \mathbb{Z}$ such that $c_{ji} \leqslant c_{ij}$ are the binary time constraints (in seconds) on the executions of β_i and β_j.

A rule of BA component construction is that every basic action β_i, with $1 \leqslant i \leqslant n$, that compose it has to have at least one time constraint of the previous form with one of the basic actions $\{\beta_1, \ldots, \beta_{i-1}\}$. This rule implies that each BA component has a maximum and a minimum duration according to the maximum and minimum solution of the underlying STP.

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. Specifically meaning that $c_{ij} = c_{ji}, \forall i, j \;\; 1 \leqslant i, j \leqslant n$.

3. Specification of needed conditions on the SP for the execution of the commands.

   Besides BA components, ARs can include *Extended Condition* (EC) components. An EC component comprise the following: a state variable identifier and a value of its domain and a duration. When the beginning time of an EC component is grounded, the EC component is satisfied if the state variable has the specified value in the time interval defined by the given beginning time and duration.

4. A way of re-use sets of commands that are usually used for the same tasks.

   An AR is defined as a simple temporal network on components (BA components and EC components). In fact, there is another possible component that can be included in a AR that is the Massive Upload Component that is used to uplink a large amount of data, but its meaning is out of the scope of the present paper.

   In the case of AR-level time constraints, the underlaying STP can be very loose to allow the needed flexibility for operation specification.

   An AR also includes time constraints used to fix a maximum and minimum beginning time for each component (as usual, the maximum and minimum time can coincide, fixing a component to a particular moment in time).

   Since components can be reused once they were defined, they provide the way of reusing sending sets of basic actions (commands).

As was told, Aquarius/SAC-D mission will be launch the next year. The mission now is on I&T phase and the basic commands for the operations of each instrument are currently being tested and debugged. The next step will be the construction of the library of Basic Commands for each payload based on the basic command library.

An example of an action request for a real-time acquisition of HSC camera is partially shown in figure 4. What can be seen there is the specification of the first basic command component that implements the camera acquisition. With the component, it is specified that a command for HKTM saving has to be executed 80 seconds before the reference time point and the command that actually implements the acquisition has to be executed 60 seconds before the reference time point (for warmup). The acquisition is indicated to be 70 seconds long.

5 Conclusion

In this paper we presented a problem that arise in some cooperative space missions. Specifically, when the planning of the operations has to be coordinated between several partners.

We enumerated several drawbacks of completely centralizing the planning by delegating this task to a unique team. We also explained how a distributed planning scheme can overcome this problems and enhance the obtained spacecraft operation plans. For this, a common and precise representation of the problem is needed. By augmenting the well-known concept of *preferences* extracted from planning with preferences frameworks, we achieved a practical way of delegating some parts of the plan to the different teams involved in a mission.

We finally presented how this ideas are applied to a space mission, the Aquiarius/SAC-D mission that is going to be launched on May of next year.

References

Fahiem Bacchus and Froduald Kabanza. Planning for temporally extended goals. *Annals of Mathematics and Artificial Intelligence*, 22(1-2):5–27, 1998.

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.

Ronen I. Brafman and Carmel Domshlak. From one to many: Planning for loosely coupled multi-agent systems. In Jussi Rintanen, Bernhard Nebel, J. Christopher Beck, and Eric A. Hansen, editors, *ICAPS*, pages 28–35. AAAI, 2008.

M. Brenner. Multiagent planning with partially ordered temporal plans. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence (IJCAI-03)*, Acapulco, Mexico, 2003. Morgan Kaufmann.

A. Cesta, S. Fratini, and A. Oddi. Planning with concurrency, time and resources: A csp-based approach. Technical report, PST@ISTC-CNR, Italy, November 2003.

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

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.

M. Ghallab, D. Nau, and P. Traverso. *Automated Planning: Theory & Practice*. Morgan Kaufmann, 2004.

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