Dynamic Allocation of Resources to Improve Scientific Return with Onboard Automated Planning

F.N. Kucinskis, M.G.V. Ferreira

Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, Brasil

Abstract

The experiments aboard the Brazilian scientific satellites are currently thought to execute in a repetitive way, collecting, storing and sending data in a cycle that does not suffer great alterations. There are, however, scientific events of randomic occurrence which demand a fast reconfiguration in the system to collect and process the data adequately. Due to the short duration of these events and the great number of states in which a system can be at the moment of the detection, the use of classical programming techniques or the ground team intervention has shown to be inefficient, and the opportunity to analyse the event is lost.

RASSO, a Resources Allocation Service for Scientific Opportunities, makes use of Artificial Intelligence Planning & Scheduling techniques to modify onboard the current plan of operations and allows an experiment to use more computational resources when a scientific event occurs, thus improving the scientific return. This paper describes RASSO, and the safe and gradual approach foreseen to implement this replanning aboard satellites.

1. Introduction

The satellites of the Brazilian Program for Scientific Satellites and Experiments run by INPE carry as payload a collection of scientific and technological experiments, which have pre-defined quotas of computational resources (memory, processor time, storage space, volume of data to transmit, etc) allocated by the onboard computer as the scientists and mission engineers had decided it, still in the phase of systems specification. As a result, the experiments collect data through a plan of operations that generally follows a repetitive pattern.

There are, however, scientific events of which occurrence, although predictable, is randomic – an ionospheric disturbance, for example, can happen at any time and last from minutes to hours. It may be important to increase the acquisition rate or the precision of the collected data to study this phenomenon. In both cases it is necessary to alter the processor time, available memory and storage space quotas dedicated to the experiments during the occurrence of the event.

Due to the short duration and the difficulty to specify exactly when an event of this kind will occur, it is not enough to leave the ground operations team in charge of the system reconfiguration. The necessary time for the event to be reported and for the ground team to send a new plan of operations to the satellite is in general much longer than the duration of the event, and in this case the opportunity to adequately analyze it will have been lost.

The need appears for allowing the experiment, when detecting a scientific event, to negotiate directly with the onboard computer the temporary allocation of the extra resources necessary to analyze it, affecting the least possible the other experiments and the proper satellite. As the number of states in which the system can be at the moment of the detection is huge, it becomes dificult the use of classical programming techniques to handle it.

In this context the Planning & Scheduling techniques, coming from the Artificial Intelligence area, is presented as a potential solution to be exploited.

2. Onboard Planning & Scheduling and the RASSO Service

According to the definition in [1], Planning is the selection and sequencing of activities in such a way that they achieve one or more goals and satisfy a set of domain constraints. Still according to [1], scheduling is the selection among alternative plans and the assignment of resources and time for each activity, so that the assignments obey the temporal restriction and the capacity limitations of a set of shared resources. In our work, scheduling is dealt as a stage inside planning, merging the domain restrictions to the ones of resources and time.

RASSO, a Resources Allocation Service for Scientific Opportunities, makes use of onboard planning & scheduling in a way to increase the satellite autonomy, allowing the system to reconfigure itself in quasi-real-time, reallocating computational resources for experiments that detect the occurrence of a scientific event and returning the system to the normal operation mode after the ending of the observation period. This service is being developed to the new onboard computer of the Brazilian scientific satellites, called COMAV (from the Portuguese acronym for "Advanced Computer").

Each experiment that communicates with the RASSO service must have a trigger condition that, when true, sends a request of resources to the onboard computer. The trigger condition indicates that a scientific event was detected and would not be adequately observed with the computational resources

available to the experiment at the moment. The request sent by the experiment informs what resources are necessary, its amount, as well as when they are needed (there is an "ASAP" option) and for how much time.

When receiving this request, RASSO directs it for its Negotiator module (as shown in Figure 1). The Negotiator first verifies if the amount of resources requested is available in the system. If so, it allocates the resources directly and informs the experiment. Otherwise, it verifies if it is possible to reallocate resources from other experiments to attend on the request.

This is made by using a set of Negotiation Rules, that indicates how much of each resource each experiment can yield, at which moment, and for how much time. An experiment can also prohibit temporarily the allocation of its resources, if it is in a critical phase of operation. This prevents it to yield resources to another one and removes it from the negotiation process. The acquired resources to attend on the request can come from more than one experiment.

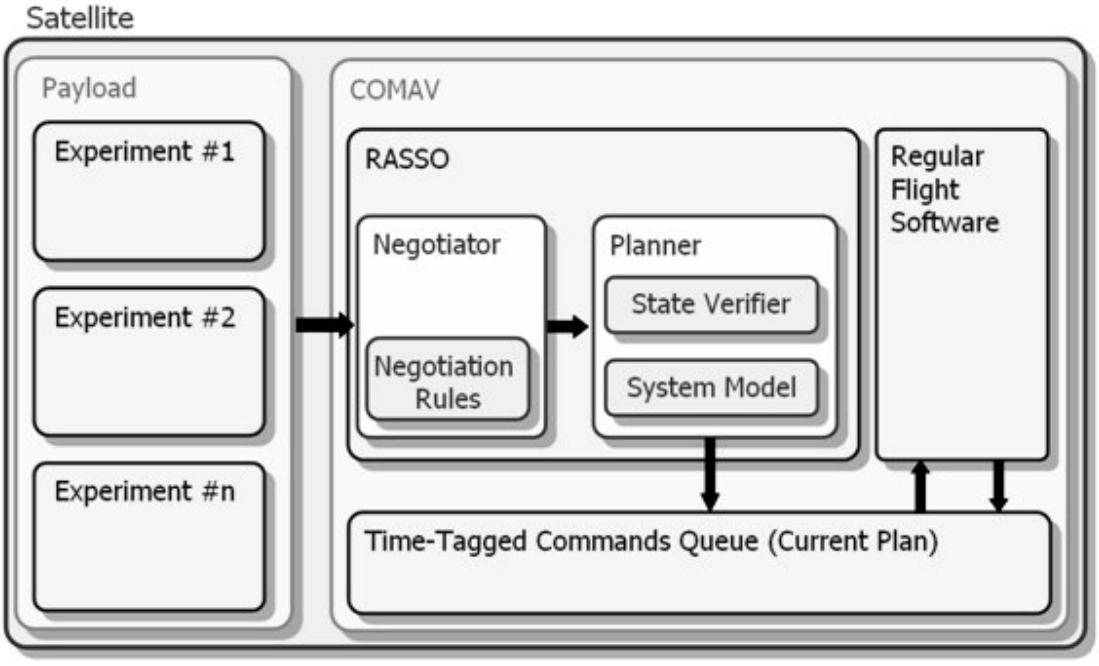

Figure 1 – The Architecture of the Service

If the Negotiator verifies that there is no free resources, but it is possible to reallocate them from other experiments, it directs the system configuration needed by this reallocation to the Planner module. This configuration is the goal state of the Planner that, on the basis of a system model, tries to create a plan of operations that reallocates the resources, keeps them to the experiment that have requested them

during the observation period and then returns them to the experiments that had yielded them originally, informs the ground control team concerning the actions taken and returns to the normal operation mode of the satellite.

After getting a plan, the Planner informs the solicitant experiment the success and directs the resulting commands to the time-tagged commands (TTC) queue of the system. If the Planner fails when trying to generate a plan or the plan process timeout is reached, the experiment is informed that its request was denied, and continues its observations in its normal operation mode.

Each experiment must foresee three operation modes: "normal", "privileged" (when it is running with more resources) and "yielding" (when it is running with less resources, yielded to a privileged resource; in this mode, the experiment data can be degradated). The first commands of the generated plan must necessarily inform the experiments affected by the reallocation to enter their new operation modes, and only then the commands of resources reallocation start to be executed. In the same way, the last commands of the plan inform the experiments to return to their normal operation modes.

3. The System Model and the State Verifier

Planning & Scheduling are generally associated to a huge volume of processing and memory consumption, which is exactly the opposite of what is found in embedded systems. COMAV, for example, will use a SPARC ERC32 processor running at 10MHz, and will have a limited amount of memory.

To deal with these limitations, RASSO works with a system model compiled with the service. Only the characteristics of the embedded software necessary for the reallocation of resources are part of the model.

As well as the proper service, the model is composed by functions and structures in C language. This prevents the use of an embedded parser to interpret a non-compiled model of the system (that generally is presented as a script), saving processing time and memory consumption. However, this also implies that any alteration or fix in the model demands the recompilation and sent to the satellite of the entire service. But as the service plus the model compiled have less than 200kb (value that we calculate at the moment), this is not a problem.

A model described in C language can also lose in clarity. Planners aimed at the space area as the NASA's ASPEN [1] and the MOIS Toolset [2] from ESA use script files with the model and procedures needed for the planning process, which clarity level cannot be obtained working directly in C.

To get around this situation a "pseudo-script" from C macros has been created. The use of these macros brings the way to represent a source file closer to the one of a script, turning the model more readable. Thus, enumerations become domains of values to attributes, structures are dealt as types of objects of the system, functions become actions of the model and so on. Figure 2 brings a simplified snippet of a pseudo-script that will be compiled with the planner in a C language header file.

```
#include "language.h"

RASSO_domain OperatingModes {normal, privileged, yielding};
RASSO_domain ExperimentID {ex_1, ex_2, ex_3, ex_4, ex_5, ex_none};

RASSO_type Experiment
{
        ExperimentID   id;
        OperatingModes mode;
}

Experiment exp1, exp2, exp3, exp4, exp5;

RASSO_action (AllocateMemory)
{
        when_planning
        {
                condition(exp1.mode == normal);

                // effects of the action in the current state
                // have to be described here
        }

        when_running
        {
                // time-tagged command(s) related to the action here
        }

        action_success;
}
```

Figure 2 – Simplified Snippet of a Model's Pseudo-Script

In the pseudo-script used to generate models to RASSO (that we named RASSO_ml, from "modeling language"), the main element is the "Action". Each action must correspond to one or more internal commands of the satellite, and is called in two moments: first, during the planning process, so that the Planner verifies what changes in the system state this action will trigger. The second call to the action happens when the plan is already generated, and the action corresponding commands must be placed in the time-tagged commands queue to be executed.

This way, the same compiled function is used when planning and when sending the plan to execution. The directives "when_planning" and "when_running" are used to distinguish the calls.

In RASSO_ml, "Types" implement classes of objects that are part of the model. These types can be instantiated as simple variables or vectors. "Domains" indicate the sets of possible values to some of the attributes of the objects. "Conditions" are used to verify if an action is applicable, given the current state (during the planning process) of the system. Conditions are also used to impose time and resource constraints.

To start the planning process the planner needs to know the state in which the system will be at the moment when it will allocate more resources for the experiment that requested them. To discover onboard a future state of the system would be a difficult task, but the RASSO architecture simplifies this. Each satellite command that affects the system model is mapped to a corresponding RASSO_action.

Then, to determine the initial state for the planning process, the planner only needs to call a State Verifier module, passing a future moment in time as parameter. The State Verifier will get the current state from the system, apply it to the model, and will discover and execute, in "planning" mode, the actions corresponding to the commands in the TTC queue that affect the model, until the last action before the given moment.

4. But How to Implement Onboard Planning Safely?

It is fact that there is a great resistance by the engineers and mission managers, justified, related to the increase of the satellites autonomy. The cost and amount of work of a space mission is in general considered big enough to trust to the satellite the taking of more decisions than that of routine, as the entrance in the emergency mode, or the attitude and orbit correction.

This resistance is even bigger when the increase of autonomy is propitiated by a technique coming from the Artificial Intelligence area. Even projects that have the intention of being testbed of technologies to increase the satellites autonomy, as ESA's PROBA [3] and missions which requirements are perfect to the use of planning, as the SWIFT observatory from NASA [4], do not implement any AI technique onboard.

So far only two missions have used onboard planning successfully: the Deep Space One probe, in 1999 [5] and the Earth Observing One satellite [6], which tests with the onboard planner started in the second

semester of 2003 and became fully operational in the first semester of 2005 (more details of both are described below, in item 6).

With so few cases of success and so big reluctance, how to convince a mission manager to implement onboard planning in his satellite?

In first place, the service must be proposed to solve a small problem, and not one that is vital to the satellite functioning. The reallocation of resources propitiated by RASSO is desirable, but if it is not possible, the experiments keep collecting and processing data in its normal operating mode.

In second, the processor cannot be fully taken by the planner during the stage of search in the space of states; this must occur in the least expensive possible way. Thus, RASSO and the system model that it uses are being developed having as goal an optimal processing. That is why no generic-purpose planner is used, but one completely focused on the solution of our specific problem.

Finally, the service implementation must be gradual and based on the increase of the confidence with respect to the results gotten. RASSO has three distinct operating modes that guarantee this: "disabled", "advice" and "act".

In the disabled mode the service is not available, and the experiments will always have its requests denied.

The advice mode exists to allow the analysis of the service reliability. In this mode the service gets the requests and always sends a refusal in reply. However, it will work in a plan as it would really attend on the request, but having a reduced priority during the planning process. The resulting plan will not be put in action but will be made available to telemetry, alongside with an operations log. Every time that a request is sent to RASSO, when in advice mode, an alert message showing that there are plans stored in the satellite waiting for analysis will be sent through normal telemetry, no matter if the gotten plan is complete or not – that is, if the planner was or was not successful in simulating the attending on the request.

The generated plan will be sent in reply to a telecommand specific for this purpose. The actions generated by the Planner will be analyzed by the operators in the ground, and compared through proper tools with the normal satellite plan, that was really executed. If it is determined that RASSO has generated a plan that will attend on the request of an experiment, still keeping the normal operation of

the satellite, this will be computed as a success of the service. On the contrary, the planning algorithm and the system model will have to be reviewed and a new version will be sent to the satellite.

Having a number of successes considered enough, the service can be set to the act mode, starting to be completely operational.

In addition to the approach described above, it must be also considered that the target application of RASSO is the scientific satellites. These are missions of lower cost, where there is more room for experimentation.

5. Current Status

RASSO is in development phase, at the same time as COMAV. The fact that both are being developed in parallel allows RASSO to make a better use of the software capabilities embedded in our next satellites, because its internal structure is being tailored as the system on which the service will run evolves.

The planning algorithm is being constantly improved, as well as the system model and state verifier. Experiment stubs are being developed to a more accurate and near to the real environment simulation. The concurrency of requests for more resources will be worked as soon as the service could fully attend on the request of one experiment each time.

We intend to have a first complete and working version of the service until the end of this year.

6. Related Work

Few projects have been made in the direction of implement onboard planning in spacecrafts, and even less were well-succeeded.

RAX-PS was the first of them. It was an onboard planner used in the NASA's Deep Space One probe [5], and, as RASSO, it is a batch planner, where the plan is executed only after the planner has generated it with completeness. The probe's thrust control and rote correction were passed to the planner for two times in May of 1999.

In October of 2003 the first tests with an onboard planner were made in the remote sensing satellite Earth Observing One (EO-1). EO-1 uses CASPER to replain activities, including downlink, based on the

observations from previous orbital cycles. It is an iterative repair planner, which proposes to detect in real time flaws that are inserted in the current plan and correct they iteratively, until there are no conflicts anymore. This way, there is always a plan ready for execution. Incremental tests had been carried until the system was considered fully operational in April of 2005 [7]. CASPER would be used in two other missions, the US Air Force's TechSat-21 and the university nanosatellite constellation Three Corner SAT [8], but the first one was canceled and the second, lost in the launch process.

CASPER is an onboard version of ASPEN, a ground-based planner used in many missions, as the proper EO-1, the Cassini probe and the Space Shuttle. It uses the ASPEN Modeling Language (AML) to describe its models. The RASSO_ml elements were inspired in the AML components.

NASA has been working in a new planning and execution architecture for onboard planning named IDEA [9]. IDEA is a framework in which the planner and execution software are combined into a "reactive planner" and operate using the same domain model.

Although onboard planning is still an area to be exploited, the use of AI planners in mission control centers to generate plans in ground that latter will be sent to the satellites is already a reality. In addition to ASPEN, referred above, NASA has developed SPIKE [10] and HSTS [11] to generate observation plans to the Hubble Space Telescope and ESA created MEXAR [12], a mission planner to the Mars Express probe.

At INPE, studies with planning have been carried in the direction of automatize the satellites operations in ground, in view of the increase of the number of satellites to be controlled that one expects for the next years. These studies were described in [13] and [14].

7. Conclusion

Given the level of development reached by onboard satellite systems, the next step to take is the increase of its autonomy. Providing technology for the use of onboard planners meets that, but it is something to be treated carefully due to the criticality of the application. To accomplish this the project proposes, allied to the technology, the use of a gradual process, based on the increasing of the confidence of the gotten results. It is intended with that to open new possibilities to be exploited for the INPE's technological and scientific experiments aboard satellites.

Through modest goals and a realistic approach, RASSO consists of a first step toward the use of onboard planning to increase the autonomy of our satellites in a near future.

8. References

[1] A. Fukunaga, G. Rabideau, S. Chien and D.Yan, "Towards an Application Framework for Automated Planning and Scheduling", In Proceedings of the International Symposium on Artificial Intelligence Robotics and Automation in Space, Tokyo, Japan, July 1997.

[2] A. Rudolph, F. J. Diekmann, A. T. Monham, F. Rother and W. Heinen, "ENVISAT Mission Automation – A First Demonstration", In Proceedings of the 6th International Symposium on Reducing the Costs of Spacecraft Ground Systems and Operations, Darmstadt, Germany, June 2005.

[3] F. Teston, R. Creasey, J. Bermyn and K. Mellab, "PROBA: ESA's Autonomy and Technology Demonstration Mission", In Proceedings of the 48th International Astronautical Congress, Turin, Italy, October 1997.

[4] J. C. Ong and M. Rackley, "Autonomous Operations for the SWIFT Mission", In Proceedings of the 7th International Conference on Space Operations, Houston, USA, October 2002.

[5] A. K. Jonsson, P. H. Moris, N. Muscettola and K. Rajan, "Planning in Interplanetary Space: Theory and Practice", In Proceedings of the 2000 Conference on AI Planning and Scheduling, pages 177-186, Breckenridge, CO, 2000. AAAI Press, Menlo Park.

[6] R. Sherwood, S. Chien, D. Tran, B. Cichy, R. Castano, A. Davies and G. Rabideau, "Operating the Autonomous Sciencecraft Experiment", In Proceedings of 8th International Conference on Space Operations, Montréal, Canada, May 2004.

[7] S. Chien, R. Sherwood, D. Tran, B. Cichy, G. Rabideau, R. Castano, A. Davies, R. Lee, D. Mandl, S. Frye, B. Trout, J. D'Agostino, S. Shulman, D. Boyer, S. Hayden, A. Sweet and S. Christa, "Lessons Learned from Autonomous Sciencecraft Experiment", In Proceedings of the Autonomous Agents and Multi-Agent Systems Conference, Utrecht, Netherlands, July 2005.

[8] R. Knight, G. Rabideau, S. Chien, B. Engelhardt and R. Sherwood, "CASPER: Space Exploration through Continuous Planning", In IEEE Intelligent Systems, September/October 2001, pp. 70-75.

[9] N. Muscettola, G. Dorais, C. Fry, R. Levinson and C. Plaunt, "IDEA: Planning at the Core of Autonomous Reactive Agents," In Proceedings of the Workshops at the AIPS-2002 Conference, Tolouse, France, April 2002.

[10] M. D. Johnson and G. E. Miller, "SPIKE: Intelligent Scheduling of Hubble Space Telescope Observations", In Intelligent Scheduling, ed. M. Fox and M. Zweben, San Francisco: Morgan-Kaufmann, 1994, pp 391-422.

[11] N. Muscettola, B. Pell, O. Hansson and S. Mohan, "Automating mission scheduling for space-based observatories", In Robotic Telescopes, ASP Conference Series vol. 79, 1995.

[12] A. Cesta, G. Cortellessa, A. Oddi, and N. Policella. "Studying Decision Support for MARS EXPRESS Planning Tasks: A Report from the MEXAR Experience", In Proceedings of the 4th International Workshop on Planning and Scheduling for Space, IWPSS'04. ESA-ESOC, Darmstadt, Germany, June 2004

[13] L. S. Cardoso, M. G. V. Ferreira and V. Orlando, "An Intelligent System for Generation of Automatic Flight Operation Plans for the Satellite Control Activities at INPE" to appear in Proceedings of the 9th International Conference on Space Operations, Rome, Italy, June 2006.

[14] A. C. Biancho, A. Carniello, M. G. V. Ferreira, J. D. S. Silva and L. S. Cardoso, "Multi-Agent Ground-Operations Automation Architecture", In Proceedings of the 56th International Astronautical Congress, Fukuoka, Japan, October 2005.