# An Onboard Knowledge Representation Tool for Satellite Autonomous Applications

Fabrício de Novaes Kucinskis
Aerospace Electronics Division (DEA)
National Institute for Space Research (INPE)
São José dos Campos, Brazil
fabricio@dea.inpe.br

Maurício Gonçalves Vieira Ferreira
Satellite Control Center (CCS)
National Institute for Space Research (INPE)
São José dos Campos, Brazil
mauricio@ccs.inpe.br

## ABSTRACT
Artificial Intelligence Planning and Scheduling (AIPS) techniques have been used both in ground-based and onboard space applications. AIPS relies on model-based domain knowledge representation systems, which is of value for many other applications, such as diagnosis systems and satellite simulators.

This paper reports the development of an autonomous satellite onboard planner developed at INPE, and how this project lead us to start creating a more structured knowledge representation tool. The tool can be used not only by a planning application, but also by diagnosis and prognosis systems, satellite simulators and more, both in onboard and ground-based environments, and even outside the space field.

## Keywords
Knowledge Representation, Artificial Intelligence Planning and Scheduling, Satellites, Autonomy.

## 1. INTRODUCTION
Low-Earth orbit artificial satellites operate most of the time without direct contact with its ground control stations. A satellite which orbits at about 750 km and has only one control station, for example, is under the immediate ground operations team supervision for just 10% of the time of each orbit it performs.

These satellites are controlled through sequences of commands sent from the ground control stations during the period in which they are in contact. In order to the satellite be able to execute the ground-generated commands at any moment of its orbit, each command has a time tag that express its execution moment. To this command sequence is given the name of operation plan.

Each command of the operation plan performs a low-level task, and it is usually necessary wide command sequences to achieve high-level goals. The generation of operation plans is a labor intensive process, carried out manually by a highly specialized staff. Many factors shall be considered in-depth: the current and future state of each satellite subsystem, time and resource constraints, the satellite attitude, orbit position and mission phase, among many others.

To reduce the effort necessary to generate operation plans and hence its costs, many space agencies have been investing in the development of computational systems that automate totally or partially the planning process. These systems are generically called planners, and some of them implement Artificial Intelligence (AI) and Operations Research (OR) techniques, known respectively as planning and scheduling. In addition to allowing the ground operations automation, these techniques can be used onboard satellites to change existing plans or create new ones in response to unpredicted situations, thus increasing its autonomy. This kind of application is relatively new, and a few cases were reported up to now.

This paper reports an onboard planner developed at the Brazilian National Institute for Space Research (INPE, in the Portuguese acronym). The lessons learned within this project had lead us to start developing a more structured knowledge representation tool, one that can be used not only by an autonomous planning application, but also by diagnosis and prognosis systems, satellite simulators and more, both in onboard and ground-based environments.

The paper is organized as follows. Section 2 presents a brief history of AI planning and scheduling applied to the ground control of space missions. Section 3 briefly describes the only two real cases of onboard planning that took place in space missions. Section 4 reports the development of an onboard planner at INPE. We describe the decisions made to handle the domain knowledge and run planning processes onboard satellites, as well as the lessons learned from such planner. Section 5 shows our current work focus, the development of an onboard domain knowledge representation tool. Section 6 lists some related work, and the current status and our final remarks are stated in Section 7.

## 2. AI PLANNING AND SCHEDULING IN SPACE MISSIONS
AI planning is the selection and ordering of activities that, when executed in a specific order, take a domain from an initial state to a desired state, or goal state. The domain is usually represented by a knowledge base in the form of a model. The set of activities that is the output of the planning process is called a plan.


*SAC'08*, March 16-20, 2008, Fortaleza, Ceará, Brazil.

Planning techniques date back from early '70s with the STRIPS planner [1], but it was the addition of OR scheduling concepts, such as resource consumption, time assignments and Constraint Satisfaction Problems (CSP), which gave planners the ability to deal with real-world problems. The union of planning and scheduling is currently known as Artificial Intelligence Planning and Scheduling (AIPS). The search for optimal or, in the worst case, complete and consistent plans that is the purpose of AIPS fits perfectly the operations routine of space missions.

NASA is the agency that has the greatest experience in the development and use of AIPS systems for mission control. After an unsuccessful demonstration of planning in early 80's [2], NASA resumed AIPS research in the 90's with the SPIKE/SPSS scheduler [3], which helps creating observation plans for the Hubble Space Telescope. SPIKE was followed by many planners, such as GERRY/GPSS [4] used for the ground maintenance plan of the Space Shuttles, HSTS, also developed for Hubble, DCAPS [5], an experiment that generated operation plans for the Shuttle's Data Chaser payload, ASPEN [6], an evolution of DCAPS used in many current missions, and finally MAPGEN [7], currently in use in the Mars Exploration Rovers mission.

ESA also has some history with AIPS. Their first experiences, also in the beginning of the '90s, were the planERS-1 [8], to generate operation plans for the ERS-1 satellite and Optimum-AIV [9] used to help the assembly, integration and verification process of equipment for the Ariane IV rockets. More recently, ESA supported a work that created the MEXAR and MEXAR II [10] planners for the Mars Express mission.

There is a work in progress at INPE to apply ground-based AIPS in its missions. This work is described in [11] and [12].

# 3. ONBOARD AIPS

In all these systems the plans are generated and validated on ground and just then sent to the spacecraft. There are cases, however, in which it is desirable the generation or change of plans aboard the spacecraft to increase its autonomy and allow it to have a quick response to external events. This is called onboard planning, and there are only two reported cases up to now, both from NASA.

In May of 1999, the Deep Space One (DS-1) probe was operated for some days through detailed operation plans generated onboard the spacecraft from high level commands sent by the ground operations team [13]. The experiment, called Remote Agent, was considered a great success.

More recently, from October of 2003, the remote sensing satellite Earth Observing One started to execute the Autonomous Sciencecraft Experiment (ASE), of which the CASPER planner, an onboard version of ASPEN, is part [14]. CASPER is responsible for replanning the satellite operations to respond to the detection of events of scientific interest, such as floods and volcanic eruptions, which increased the scientific return. The ASE implementation was gradual, and in April of 2005, the ground operations team was already using it in normal tasks [15].

These missions place NASA as the only agency to use AIPS aboard its spacecrafts. ESA has also been investing in the increase of autonomy with projects such as PROBA [16], but with no onboard planning up to now.

# 4. RESOURCES ALLOCATION SERVICE FOR SCIENTIFIC OPPORTUNITIES

Apart its work with ground-based AIPS, INPE is also developing onboard planning technology. The Resources Allocation Service for Scientific Opportunities (RASSO) is an onboard replanning service which goal is to change scientific satellites' ground-generated operation plans in such a way to reallocate resources (mass memory and power, for example) to experiments when they detect the occurrence of short-duration scientific phenomena. Operating with more resources than originally programmed, the experiments can do a better observation of such phenomena.

Figure 1 shows the RASSO architecture. The arrows indicate the data flow between the modules during the planning process. Follows a brief description of the service functioning.

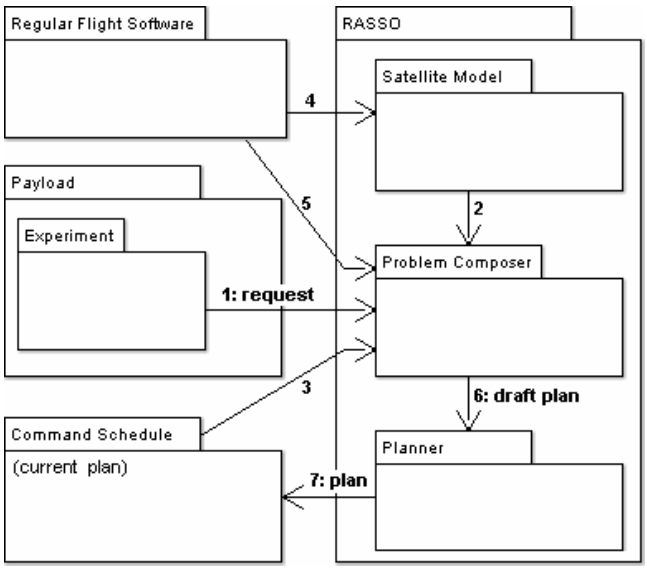


**Figure 1 - The RASSO Architecture**

An experiment that detects the occurrence of a short-duration phenomenon sends a request for more resources to RASSO, to make a better observation (arrow number 1). When receiving the request, RASSO composes a well-defined problem in the form of a draft operation plan. This draft plan is created consolidating information from several sources (arrows 2 to 5), and then is directed to the planner module (arrow 6), responsible for working out the conflicts that were inserted in the operation plan because of the request from the experiment, respecting a set of constraints and goals that were imposed to it. When succeeding in creating a plan that takes care of all these requirements, RASSO sends it to the satellite's command schedule, turning it the new current operation plan (arrow 7).

The next section describes how the knowledge about the domain is embedded in the satellite model to be used by the planner.

## 4.1 Knowledge Representation in RASSO

RASSO runs on an ERC32 RISC processor (SPARC 32 bits architecture) at 12 MHz, and has less than 1 Mbyte of memory available to its execution thread. In order to make the planner

search process feasible with this limited computational power, we made two main decisions.

The first one was about the kind of problem representation and search algorithm to be used in such onboard application. The planning problem should be represented as a Constraint Satisfaction Problem (CSP) in the form of a draft plan (as described in section 4), and a local search algorithm, guided by scheduling constraints, should be used to solve it. When necessary, disturbances should be inserted in the plan to escape from local minima and plateau regions of the search space.

The second decision was about how to describe the satellite domain model to be used by the planner. We concluded that the satellite model should be described in the same language in which the planner would be developed, the C++ programming language. The model elements would be translated in compile-time to data structures in which they would be handled.

The C++ language is capable of describing a model adequately, but a model described in C++ would lose in clarity. An engineer or scientist not used with the programming language and with the software structure would not understand what is being represented. Thus, to allow the direct storage in the right data structures and still keep the model readable, we decided to create a model description language over C++, through the use of macros that hide all structures, pointers and function calls used by the planner.

The use of macros to implement this language, that was named RASSO_ml, makes the task of converting a model instruction to data structures being a responsibility of the C++ compiler pre-processor. The model description is comprised in a "domain_model.h" header file, which is compiled together with the planner. Thus, the model elements are always ready to be manipulated by the planner, with no parsing and eliminating a great part of the initialization process.

This approach is similar to the one taken in the Task Description Language (TDL) [17]. The difference is that, in TDL, it is not the compiler pre-processor itself that translates the language constructs into programming code, but a specialized parser.

With RASSO_ml it is possible to describe model components through classes and elements (instances of classes), activities with its pre-conditions and effects, exogenous events, resources and constraints. Activities and events can take advantage of any C++ language construct, such as conditionals, switch statements and loops, as well as C++ libraries such as <math.h>. There is no space in this paper to describe all this features, so see [18] for more information.

## 4.2 Lessons Learned

When we started RASSO, we were not totally sure that a domain description mixed with programming source code would work as expected, but this proved to be a right choice. Any domain specialist familiarized with languages such as C, C++ or Java can quickly edit models in RASSO_ml, being necessary only to learn some basic concepts and a few language instructions. The handling of the compiled model components by the planner is pretty simple, since the same instructions that describe activities and events effects can be called by the planner to get or set the model state – even the initial and goal states.

The local search algorithm with CSP has shown to be appropriate for onboard execution, although we have room for improvements. As an example, currently there is no plan optimization. Any consistent plan, which leads the satellite to the goal state and respects all the imposed constraints, are accepted.

When showing RASSO results, we were asked by colleagues if this system could be used in a range of different applications, such as onboard diagnosis and prognosis and satellite simulation. After some discussion, we figured out that it is not RASSO which would be used in those applications, but the domain model it comprises. So, we decided that the next step would be to develop a more structured knowledge representation tool, over which different onboard applications, and even ground ones, such as a satellite simulator, could be made. We called this tool the Space System Model (SSM).

# 5. THE SPACE SYSTEM MODEL

SSM is the evolution of RASSO's satellite domain model. It provides a domain representation language, state inference, constraint propagation and resource consumption/generation profiles, among other features.

Domain knowledge is represented by two different descriptions: a static (structural) description and a dynamic (behavioral) description. A real-world interface maps the model elements, resources, activities and events to real satellite subsystems, commands, etc, allowing SSM to interact with the satellite. Figure 2 shows the SSM knowledge representation structure.

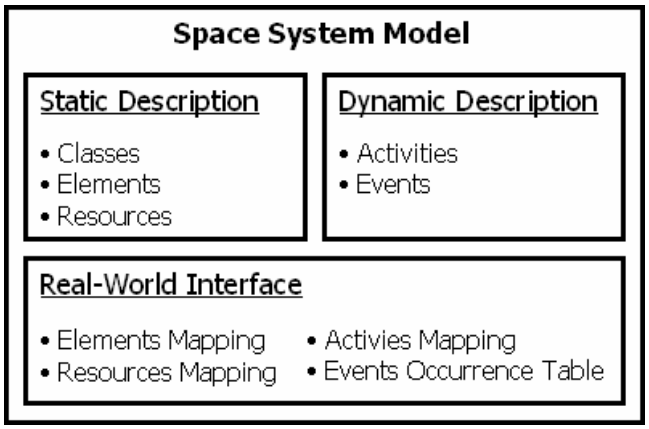


**Figure 2 - SSM Knowledge Representation Structure**

The static description contains the satellite model structure, that is, the elements that constitute a satellite (such as its subsystems and payload), the classes from which the elements are created, and the resources available for consumption by those elements. It is those description components that are translated by the C++ compiler into data structures to be handled by SSM in run-time.

Elements are instances of classes, and a timeline is created for each element. That means that SSM provides methods to get or set the initial, previous, current and goal states of an element.

Resources consumption/generation are informed in rates, not in "closed" amounts. When calling an activity that turns on a thruster, for example, one can inform that this thruster consumes fuel "at the rate of 0.1 units per second". A following turn off activity will cease the fuel consumption. This is closer to the real

satellite operation and gives more flexibility to search algorithms than the usual planning approach, in that one would state that "the thruster will operate for ten seconds and will consume 1 unit of fuel".

The dynamic description contains the operators that can change the model state. These operators are called by applications at run-time to infer the satellite behavior. There are two types of operators: activities and events. Activities are related to any kind of internal satellite command, at any level. It can represent a low-level single command, or an entire onboard procedure (this is up to the modeler). Events describe the effects of exogenous events, such as the entrance in eclipse, over the satellite.

The model is described in the same way it was in RASSO, that is, in a C++ header file, with a new language based on RASSO_ml. A simplified code snippet is shown in Figure 3.

```
using namespace ssm;

// static description ///////////////////////
create_class(experiment,
               exp_name  name;
               bool      on;
               int       sample_rate;
            );

create_element(ionex, experiment);
create_element(grom,  experiment);


// dynamic description //////////////////////
activity turn_on(experiment exp)
{
    // activity pre-conditions
    precondition(exp.on.get_current()    ↵
                                  == false);

    // ionex experiment cannot be turned off
    precondition(exp.name != ionex);

    // activity effects
    exp.on.set_current(true);

    // report success to the SSM
    activity_completed;
}
```

**Figure 3 – A simplified domain_model.h code snippet**

One of the uses of the real-world interface is the SSM model check feature. It consists of inferring the satellite response to the current operation plan and exogenous events, and verify the expected behavior at given intervals. It is important to use this feature to verify the accuracy of the model, before using it in a planning or diagnosis application.

Another important part of SSM are the constraints. They are informed inside the activities and events, or submitted to SSM by an application (for example, a planner could submit constraints alongside with goals). Constraint propagation methods are applied to the model at any change in its state.

Two of the applications foreseen to be developed over SSM are outlined in the following subsections.

## 5.1 Onboard Autonomous Reaction Agent

The first application to use SSM will be the Onboard Autonomous Reaction Agent (OAR Agent). Roughly speaking it's a new version of RASSO, with a wider scope – it is not intended just to reallocate resources to experiments, but to modify plans at any contingency situation to achieve different kinds of goals.

The search algorithm will be an improved version of the one used in RASSO. In an ERC32 processor running at 12MHz, RASSO is capable of finding consistent plans in about two minutes. That is far from being a bad result, but it could be better. We realized that we can develop reasonable search algorithms, but this is not our strong point. That's why OAR agent will accept third-party planning algorithms, that implement theirs own search strategy and heuristics.

## 5.2 Diagnosis and Prognosis Agent

An extension of the SSM model check feature, the Diagnosis and Prognosis Agent will infer the satellite behavior in the short and medium-term future. It will delivery commands or send reports to the ground operations team when specific (probably error) states are predicted, or when one predicted state differs from the actual value that is read from the satellite.

We have plans to use this agent also in a ground-based satellite simulation system.

# 6. RELATED WORK

NASA has some onboard model-based systems already used in space missions, and others in development.

The DS-1 Remote Agent experiment used a tool called Livingstone [19] to keep its onboard models. Livingstone is developed in LISP and is capable not only to infer the overall behavior of the system, but also to monitor it. Since the Remote Agent, Livingstone has flown in a number of different missions.

The Remote Agent is also the basis of another system that relies on an onboard model: the Intelligent Distributed Execution Architecture (IDEA) [20], a framework that unifies planning and execution and is under development by NASA Ames Research Center. IDEA is proposed to use the same primitives and semantic to the model in all system layers, to simplify its implementation and validation. The model and all the states it assumes are kept in the Plan Service Layer (PSL) database, which is similar to our Space System Model.

Another onboard planning and execution system that is being developed is the Coupled Layered Architecture for Robotic Autonomy (CLARAty) [21]. CLARAty is a joint project between NASA JPL, NASA Ames, Carnegie Mellon University and other universities. It is meant to be used in space-based robotic control applications. CLARAty has two layers: a functional layer of robotic primitives, coupled with a decision layer of planning and execution functionality. The result is that it depends on two modeling frameworks, one for planning (CASPER), and another for execution (TDL). SSM applications, on the contrary, will rely on only one domain model.

# 7. FINAL REMARKS

The Space System Model is currently under development. Once we have finished this phase, we will start developing the applications described in Section 5, being the OAR Agent the first

of them. SSM has also a great potential for being used in non-space missions. We plan to explore those fields, once we have some results with our space applications.

## 8. ACKNOWLEDGMENTS

We would like to thank the Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), for supporting part of this work.

## 9. REFERENCES

[1] Fikes, R. E. and Nilsson, N. J. STRIPS: a new approach to the application of theorem proving to problem-solving. *Artificial Intelligence, v. 2, n. 3-4*, pp. 189-208, 1971.

[2] Vere, S. Planning in time: windows and durations for activities and goals. *IEEE Transactions on Pattern Analysis and Machine Intelligence, v. 5*, pp. 246-267, 1983.

[3] Johnston, M. SPIKE: AI scheduling for NASA's Hubble space telescope. In *Proceedings of the IEEE conference on AI applications*, 1990, Santa Barbara, USA, pp. 184-190.

[4] Zweben, B., Davis, M., Daun, E. and Dale, M. Scheduling and rescheduling with iterative repair. *IEEE Transactions on Systems, Man, and Cybernetics, v. 23, n. 6*, Nov / Dec 1993.

[5] Rabideau, G., Chien, S., Mann, T., Willis, J., Siewert, S. and Stone, P. Interactive, repair-based planning and scheduling for shuttle payload operations. In *Proceedings of the IEEE aerospace conference*, 1997, Aspen, CO, USA.

[6] Fukunaga, A., Rabideau, G., Chien, S. and Yan, D. Towards an application framework for automated planning and scheduling. In *Proceedings of the international symposium on artificial intelligence robotics and automation in space (I-SAIRAS)*, 1997, Tokyo, Japan.

[7] Ai-Chang, M., Bresina, J., Charest, L., Chase, A., Cheng-Jung, J., Jónsson, A., Kanefsky, B., Morris, P., Rajan, K., Yglesias, J., Chafin, B., Dias, W. and Maldague, P. MAPGEN: mixed-initiative planning and scheduling for the Mars Exploration Rover mission. *IEEE Intelligent Systems, v. 19, n. 1*, pp. 8-12, Jan/Feb 2004.

[8] Fuchs, J. J., Gasquet, A., Olalainty, B. and Currie, K. W. PlanERS-1: an expert planning system for generating spacecraft mission plans. In *Proceedings of the international expert planning systems conference*, 1990, London, pp70-75.

[9] Aarup, M., Arentoft, M. M., Parrod, Y., Stader, J. and Stokes, I. Optimum-AIV: a knowledge-based planning and scheduling system for spacecraft AIV. *Knowledge Based Scheduling,* Fox, M. and Zweben, M., ed.. Morgan Kaufmann, San Mateo, CA, USA, 1994.

[10] Cesta, A., Oddi, A., Cortellessa, G., Fratini, S. and Policella, N. AI-based tools for continuous support to mission planning. In *Proceedings of the 9th international conference on space operations (SpaceOps 06)*, Rome, Italy, June 2006.

[11] Biancho, A. C., Carniello, A., Ferreira, M. G. V., Silva, J. D. S. and Cardoso, L. S. Multi-agent ground-operations automation architecture. In *Proceedings of the 56th international astronautical congress (IAC '05)*, Fukuoka, Japan, October 2005.

[12] Cardoso, L. S., Ferreira, M. G. V. and Orlando, V. An intelligent system for generation of automatic flight operation plans for the satellite control activities at INPE. In *Proceedings of the 9th international conference on space operations (SpaceOps '06)*, Rome, Italy, June 2006.

[13] Bernard, D., Dorais, G., Gamble, E., Kanefsky, B., Kurien, J., Man, G. K., Millar, W., Muscettola, N., Nayak, P., Rajan, K., Rouquette, N., Smith, B., Taylor, W. and Tung, Y. W. Spacecraft autonomy flight experience: the DS1 remote agent experiment, In *Proceedings of the AIAA 1999*, Albuquerque, NM, USA, September 1999.

[14] Chien, S., Sherwood, R., Tran, D., Castano, R., Cichy, B., Davies, A., Rabideau, G., Tang, N., Burl, M., Mandl, D., Frye, S., Hengemihle, J., D'agostino, J., Bote, R., Trout, B., Shulman, S., Ungar, S., Van Gaasbeck, J., Boyer, D., Griffin, M., Burke, H., Greeley, R., Doggett, T., Williams, K., Baker, V. and Dohm, J. Autonomous science on the EO-1 mission. In *Proceedings of the international symposium on artificial intelligence robotics and automation in space (I-SAIRAS)*, 2003, Nara, Japan.

[15] Chien, S., Sherwood, R., Tran, D., Cichy, B., Rabideau, G., Castaño, R., Davies, A., Mandl, D., Frye, S., Trout, B., D'Agostino, J., Shulman, S., Boyer, D., Hayden, S., Sweet, A. and Christa, S. Lessons learned from autonomous sciencecraft experiment. In *Proceedings of the autonomous agents and multi-agent systems conference*, Utrecht, Netherlands, July 2005.

[16] Teston, F., Creasey, R., Bermyn, J., Bernaerts, D. and Mellab, K. PROBA: ESA's autonomy and technology demonstration mission. In *Proceedings of the 13th AIAA/USU conference on small satellites*, Logan, UT, USA, September 23-26, 1999.

[17] Simmons, R., Apfelbaum, D. A task description language for robot control. In *Proceedings of Conference on Intelligent Robotics and Systems*, 1998, Vancouver, Canada.

[18] Kucinskis, F. N. and Ferreira, M. G. V. Dynamic allocation of resources to improve scientific return with onboard automated replanning. In *Space operations: mission management, technologies, and current applications*, Progress in Astronautics and Aeronautics, v. 220, Chapter 20, pp. 345-359, AIAA, Reston, USA, September 2007.

[19] B. C. Williams and P. P. Nayak. A model-based approach to reactive self-configuring systems. In *Proceedings of AAAI*, 1996.

[20] 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.

[21] I.A. Nesnas, A. Wright, M. Bajracharya, R. Simmons, T. Estlin and Kim, W. S. CLARAty: an architecture for reusable robotic software. In *Proceedings of the SPIE Aerosense Conference*, Orlando, Florida, April 2003.