# Taking the ECSS Autonomy Concepts One Step Further

Fabrício de Novaes Kucinskis and Maurício Gonçalves Vieira Ferreira
*National Institute for Space Research (INPE), São José dos Campos, Brazil*

**This paper presents the first efforts made at the Brazilian National Institute for Space Research (INPE) to increase the autonomy of its satellites through the adoption of model-based reasoning on the flight software.**

**A prototype for an on-board, goal-oriented replanning system was developed. With the lessons learned from this prototype, we are implementing an on-board service that provides states inference, which can be used as an autonomy kernel by autonomous applications.**

**But instead of creating one more brand new, one-of-a-kind system, we are trying to make this service fit into the European Committee for Space Standardization (ECSS) standards, which have been adopted by INPE. ECSS has not only defined an autonomy concept for space missions, but has also divided it in categories and levels.**

**We identified a gap between two of the last autonomy levels, and propose a new intermediate level to fill it. Our Internal State Inference Service (ISIS) implements this new level.**

## I. Introduction

It is a general agreement among the space community that spacecrafts have to be increasingly more autonomous. A survey[1] on NASA's command execution systems highlights that "as NASA's missions become more complex, autonomy becomes increasingly important to the success of those missions." A paper about ESA's PROBA satellite[2] states that "it is generally expected that, in the near future, mission operations activities will evolve from ground-based control towards on-board monitoring and control", and that "more complex functions may gradually be migrated to the on-board computer. On-ground operations spacecraft activities by humans will essentially be limited to the initialization phase (establishing the routine operations) and to emergency support".

But the transfer of new functions to the on-board computer is a hard task. It involves the capturing of a large amount of knowledge from human experts, the filtering of the resulting knowledge base to keep just what is needed for the mission to be accomplished, and the representation of that knowledge in a concise and unambiguous form. Moreover, the flight software (FSW) has to operate over the knowledge base with scarce on-board computational resources, and make decisions in such a way that the mission is not endangered, nor its performance compromised.

That's why, except for natively closed-loop tasks like attitude control, only few missions have flown some kind of autonomous FSW – almost all of them for limited periods of time, to perform one-shot technology validation experiments. However, space agencies such as NASA, ESA and CNES keep carrying research and developing prototypes for autonomous systems.

The Brazilian National Institute for Space Research (INPE, in the Portuguese acronym) is also trying to address these problems. We have developed a prototype for a model-based autonomous replanning FSW, and this first experience allowed us to try a wider approach, an 'autonomy kernel' in the form of an on-board service. This service provides states inference, resources profiling, constraint propagation and simple temporal networks features for on-board autonomous applications that perform tasks such as diagnosis/prognosis and goal-oriented mission replanning.

But to make this service ready to flight in our future satellites, we decided to avoid the development of one more brand new, one-of-a-kind system and are trying to make it fit into the European Committee for Space Standardization (ECSS) engineering standards, which have been adopted by INPE.

This paper presents our work and how it relates to the ECSS standards. Chapter II has a compilation of the ECSS autonomy concepts and the related engineering standards. Chapter III shows a survey on the levels of autonomy reached by the current INPE missions. The efforts that have been made at INPE to increase the FSW autonomy for its future satellites are reported on Chapter IV. Chapter V brings our proposition for an extension of the ECSS autonomy levels. An Internal State Inference Service that meets this proposition is described in Chapter VI, followed by the current state of this work and our final remarks, in Chapter VII.

# II. The ECSS Autonomy Concepts and Related Standards

The European Committee for Space Standardization was created in 1993, replacing ESA as the European authority responsible for the creation and publishing of standards for space projects. The ECSS members are ESA, European national space agencies such as ASI, CNES, DLR and others, Eurospace representing the European industry and the Canadian Space Agency (CSA).

Since its creation, ECSS has published dozens of standards, organized in three main branches: Space Project Management, Space Product Assurance and Space Engineering. In this Chapter we briefly describe three of the Engineering standards directly related to on-board autonomy, and that are relevant to the work presented here.

## A. Spacecraft Segment Operability

This standard[3] defines sets of requirements for the design of on-board functions for unmanned space segments. Those requirements were selected in order to ensure that the space segment can be operated in-flight in any nominal or predefined contingency situation.

Among other things, the standard presents the ECSS concept for on-board autonomy:

*"Capability of the space segment to manage nominal or contingency operations without ground segment intervention for a given period of time."*

The standard also translates this concept into something manageable at the system level. There is a chapter dedicated to the on-board autonomy requirements, which specify three branches for autonomy:

1. Autonomy for execution of nominal mission operations (branch 'E'),
2. Autonomy for mission data management (branch 'D') and
3. Autonomy for on-board fault management (branch 'F').

But dividing autonomy in categories seems to be not enough to properly manage it. As Jónsson *et al.* stated[4], "autonomy is not an all-or-nothing characteristic of a system, but something that admits of degree". In accordance with this, each category has a set of autonomy levels that specify how much autonomy is needed for a given mission. The next Chapter will treat the autonomy levels in more detail.

Each branch has also its own 'autonomy duration', i.e., a mission-constant value that defines for how long the space segment has to continue operating without intervention from the ground segment.

## B. Ground Systems and Operations – Telemetry and Telecommand Packet Utilization

This standard[5], commonly referred to as 'PUS' (from its previous ESA version, the Packet Utilization Standard), defines the application-level interface between the ground and space segments. It relies on the CCSDS protocols for data transfer and describes a number of operations concepts that are satisfied by a set of services.

These services comprise almost all common operations to control and monitor the space segment. In this context, the space segment is the service provider, while the ground segment is the 'main' consumer – on-board applications can also consume services. The interface between the segments is defined in terms of packets requests (telecommands) and reports (telemetry).

The adoption of PUS for a given mission promotes the gradual increase of the on-board autonomy. Its implementation starts from minimal sets of capabilities, which have only the basic for the spacecraft operation, to extended sets of capabilities that transfer to the spacecraft advanced functions such as the conditional execution of interlocked time-tagged commands and even on-board procedures. It is up to the mission designers to define what capabilities to implement for each service.

In 2002, Merri *et al.*[6] informed that "virtually all ESA missions are considering the adoption of PUS". Following this trend, the Brazilian missions that are based on INPE's Multi-Mission Platform also implement PUS.

## C. Ground Systems and Operations – Monitoring and Control Data Definition

The purpose of this standard[7] is to define the data to be provided by a supplier entity (a company, for example) to a customer in a given contract for a space mission, in order to guarantee that the customer will be able to monitor and control the product to be delivered.

Although, in principle, this seems not to be related with on-board autonomy, this standard defines two main concepts for the automation of a space mission: the Space System Model and the Domain-Specific Views.

A Space System Model (SSM) is a:

*"Representation of the space system in terms of its decomposition into* ***system elements****, the* ***activities*** *that can be performed on these system elements, the* ***reporting data*** *that reflects the state of these system elements and the* ***events*** *that can be raised and handled for the control of these system elements, activities or reporting data."*

The SSM is a means to capture the knowledge about a space system. It is intended to be a textual and graphical description of the entire mission, covering all its segments.

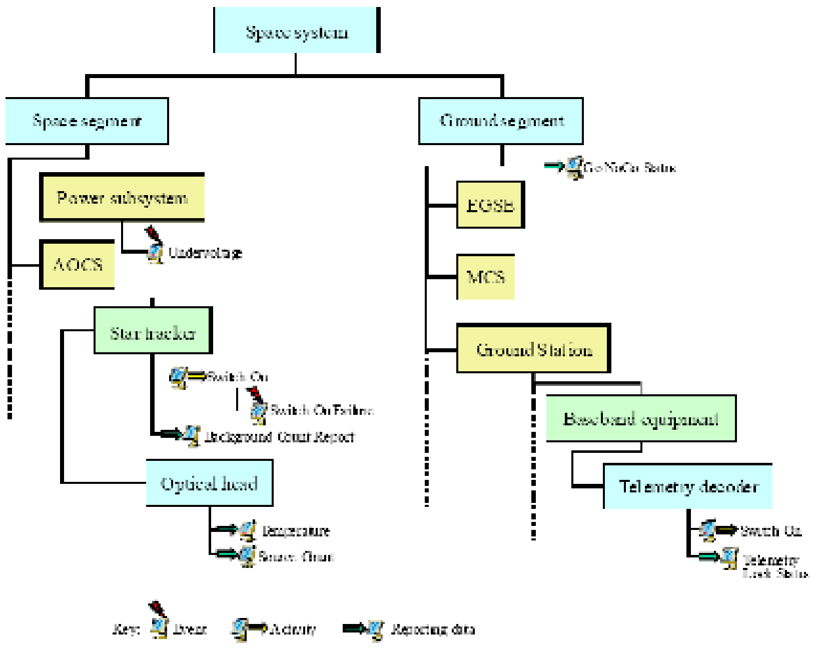

**Figure 1. Example of a Space System Model (from Ref. 8).**

A Domain-Specific View is a sub-set of the SSM that is relevant for a given application. In the Figure 1 example, a Domain-Specific View for an attitude control simulation software could have as root the AOCS system element.

The SSM and the Domain-Specific View are important concepts when one implements an on-board model-based autonomous FSW, as we will show in Chapter VI.

## III. A Survey on the Levels of Autonomy of the Current INPE Missions

The first step on the increase of on-board autonomy in accordance to the ECSS standards is to determine what levels are being reached by our current projects. So, we did a survey on the autonomy levels of INPE's next satellites, currently in different development phases: CBERS 3 and 4, Amazonia-1 and Lattes.

The CBERS program is a long-term cooperation between INPE and the China Academy of Space Technology (CAST), from which three remote sensing satellites have already been launched. The next two satellites, CBERS 3 and 4, are scheduled for launch in 2011 and 2013, respectively.

The Amazonia-1 and Lattes satellites are both based on the Multi-Mission Platform (MMP), a common satellite bus for 500-kg class satellites, on which different payloads can be attached. Amazonia-1 is a remote sensing satellite equipped with an AWFI camera for the monitoring of the Brazilian Rain Forest. It is scheduled for launch in 2013. Lattes is a scientific satellite that will perform experiments both on the space climate and x-rays monitoring. Both Amazonia-1 and Lattes implement the ECSS PUS, although in different configurations.

The following tables summarize the results of our survey.

| Autonomy Levels for Mission Data Management | | | | | |
|---|---|---|---|---|---|
| **Level** | **Description** | **Functions** | **CBERS 3 and 4** | **Amazonia-1** | **Lattes** |
| D1 | Storage on-board of essential mission data following a ground outage or a failure situation. | • Storage and retrieval of event reports;<br>• Storage management. | yes | yes | yes |
| D2 | Storage on-board of all mission data, i.e. the space segment is independent from the availability of the ground segment. | • As D1 plus storage and retrieval of all mission data. | no | no | no<br>(TBC) |

With respect to mission data management, all of the studied missions are in level D1. To reach level D2, it is necessary to augment the resources available for the space segment, mainly storage memory and downlink rate. This would involve considerable cost and engineering effort, but does not constitute a FSW problem.

| Autonomy Levels for On-Board Fault Management | | | | | |
|---|---|---|---|---|---|
| **Level** | **Description** | **Functions** | **CBERS 3 and 4** | **Amazonia-1** | **Lattes** |
| F1 | Establish safe space segment configuration following an on-board failure. | • Identify anomalies and report to ground segment;<br>• Reconfigure on-board systems to isolate failed equipment or functions;<br>• Place space segment in a safe state. | yes | yes | yes |
| F2 | Re-establish nominal mission operations following an on-board failure. | • As F1, plus reconfigure to a nominal operational configuration;<br>• Resume execution of nominal operations;<br>• Resume generation of mission products. | yes | yes | yes |

The ECSS autonomy levels for on-board fault management are quite common, so we were not surprised by all missions reaching level F2.

| Autonomy Levels for Mission Nominal Operations Execution | | | | | |
|---|---|---|---|---|---|
| **Level** | **Description** | **Functions** | **CBERS 3 and 4** | **Amazonia-1** | **Lattes** |
| E1 | Mission execution under ground control; limited on-board capability for safety issues. | • Real-time control from ground for nominal operations;<br>• Execution of time-tagged commands for safety issues. | yes | yes | yes |
| E2 | Execution of pre-planned, ground-defined, mission operations on-board. | • Capability to store time-based commands in an on-board scheduler. | yes | yes | yes |
| E3 | Execution of adaptive mission operations on-board. | • Event-based autonomous operations;<br>• Execution of on-board operations control procedures (OBCPs). | no | partial<br>(no OBCPs) | partial<br>(no OBCPs) |
| E4 | Execution of goal-oriented mission operations on- | • Goal-oriented mission replanning. | no | no | no |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
|  | board. |  |  |  |  |

For nominal operations execution, all missions reach level E2. The MMP-based satellites also have event-based command executions, which put them partially in level E3. As ECSS has not yet published a final version of its standard ‘Spacecraft On-board Control Procedures’[9], the implementation of OBCPs is not planned.

The operations execution branch is clearly the one that has more work to be done, and the one that most impact the mission. The E4 is the highest level for any of the autonomy branches defined by ECSS. What called our attention here is that the gap between it and the previous level is notably bigger than all of the others. Shouldn’t be an intermediate level before E4? We return to this question in Chapter V.

## IV. Efforts for Increasing the Satellite's FSW Autonomy at INPE

Before trying to make our efforts fit into the ECSS concepts and hence turn the final product into something ready to flight in our future satellites, we have developed at INPE a prototype for an on-board, goal-oriented replanning software.

NASA projects have shown that the most promising technology to achieve on-board replanning is the model-based reasoning. Examples of such projects are RAX[10], ASE[11], IDEA[12] and CLARATy[13]. Our prototype also followed this approach, and was called the Resources Allocation Service for Scientific Opportunities (RASSO).

RASSO is capable of changing the operation plan of a scientific satellite in order to reallocate resources (i.e. power and memory) to experiments that detect the occurrence of short-duration phenomena. Operating with more resources than originally programmed, the experiments can run for more time and store more data, performing a better observation of the phenomenon.

RASSO comprises a model of the experiments that describes its operation logic, resources consumption and the activation/deactivation schedule. This description is made in a modeling language developed over C. This allows the domain expert to use, in the modeling process, programming language features such as conditionals, loops and the math library, to better describe the behavior of the satellite. The model is compiled and linked with the other RASSO modules, the Problem Composer and the Planner.

The problem representation is based on Constraint Satisfaction Problems (CSP) and the planner uses a simple local search algorithm with backtracking. On an ERC32 processor running at 12 MHz, the RASSO response time, from the reception of a request for more resources to the sending of a new operation plan to the command schedule, is on the order of dozens of seconds.

RASSO can be seen as our first attempt to reach the autonomy level E4. It gave us a clear idea of the main difficulties and techniques for managing on-board FSW autonomy. For more information on RASSO, see Ref. 14.

Among the lessons learned with this prototype, we concluded that our development philosophy should be aligned with the one currently in use by INPE, in order to better integrate an autonomous on-board application with the rest of the FSW, and with the missions operations in general. This resulted in our current adherence to the ECSS engineering standards.

Also, we understand that to directly bring such a level of autonomy to the space segment of a real mission is a too big step. A gradual approach, in which the main components could be put to execution one by one, would be more adequate. In this case, the validation of the on-board model shall be the first one.

## V. Filling the Gap between the ECSS Autonomy Levels

For the MMP-based satellites, that implement the ECSS PUS, the next autonomy level to reach would be E4. This level foresees the execution of goal-oriented mission operations on-board, that implies in on-board replanning.

As NASA has shown, model-based reasoning is the most promising technology to achieve this. However, as previously stated, there is the need to gradually implement and validate the on-board models, the replanning algorithm and all of the interfaces with the rest of the FSW and ground segment.

A more reasonable approach would be to first apply the model-based reasoning technology to an on-board monitoring application that would give us the opportunity to validate the model and learn how to work with it. There would be no need to worry with problems as the optimization of a search space algorithm or with the impact of the autonomous changes over the ground-generated command sequences. This validated and well-known model could then be used by applications such as on-board replanning and diagnosis[*]. With such validation done, one would be

[*] NASA has also good examples of model-based, on-board diagnostic software: Livingstone[15] and its successors, L2[16] and HyDE[17].

half way to achieve the last ECSS autonomy level.

But ECSS hasn't predicted such an intermediate level, on which the base for level E4 could be made and validated through a real application. So, we propose a new autonomy level, shown in the following table:

| **Autonomy Levels for On-Board Fault Management** ***(new proposed level)*** | | |
|---|---|---|
| **Level** | **Description** | **Functions** |
| F3 | Predict potential faults and warn the ground segment. | • Constant monitoring for potential future faults based on a model of the space segment;<br>• Generation of warnings for the ground segment when a potential fault is detected;<br>• On-board model checking and update. |

This new fault management autonomy level gives to the space segment not only the capability to autonomously react to faults, as stated by the previous levels, but also to predict them and warn the ground segment. The capability to take an autonomous action in case of the prediction of a potential fault belongs to the operations execution level E3. This proposed level F3 complements the on-board fault management branch of the ECSS autonomy concept, and pave the way to level E4. Next chapter shows what we are doing to reach this level.

## VI. The Internal State Inference Service

The ECSS PUS groups sets of capabilities into on-board services that can be consumed both by the ground and space segments. If one wants to bring to the space segment a new set of capabilities related to the operation of an on-board model, it is straightforward to do so through a new on-board service. So, we have created the Internal State Inference Service, ISIS.

ISIS is the evolution of the RASSO satellite model, adapted to reflect the ECSS concepts. It provides to on-board applications and to the ground segment features such as state inference, simple temporal networks, constraint propagation and resources consumption profiles.

The on-board model is a Domain-Specific View of a complete Space System Model (SSM) that is kept on ground. The modeling components were adapted to match the ones defined in ECSS-E-ST-70-31C: system elements, activities, events and reporting data. Figure 2 shows the on-board model structure and its interfaces with the actual satellite data. We detail the model structure on the following item.

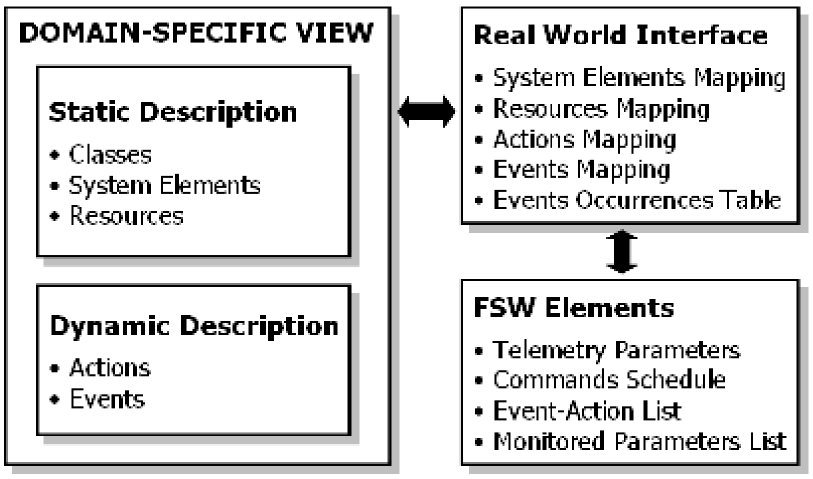

**Figure 2. The on-board model and its interface with the FSW.**

### A. The On-board Domain-Specific View

The domain knowledge contained in ISIS is represented by two complimentary descriptions: a static (structural) description and a dynamic (behavioral) description.

The static description contains the satellite model structure, i.e., the system elements that constitute a satellite (such as its subsystems and payload), the resources available for consumption by those elements, and the classes from which both the system elements and resources are instantiated.

The dynamic description contains the operators that can change the model state. There are two types of operators: actions[†] and events. The actions are directly mapped to satellite commands that can be stored in the command schedule. Events describe the effects of the occurrence of exogenous events over the satellite, such as the entrance in eclipse.

Resources consumption/generation is informed in rates, not in 'closed' amounts. When calling an action that turns on a thruster, for example, one can inform that the thruster will consume fuel 'at the rate of 0.1 units per second'. A following turn off action will cease the fuel consumption. This is different from most planning and state inference systems, where is common to have activities with an associated duration. This approach is also 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'.

Having evolved from RASSO, ISIS models are now object-oriented, described with a modeling language developed over C++.

## B. The Real World Interface

A Real World Interface module maps each of the model components to real satellite subsystems, commands, etc. The basic system elements (the ones that are not decomposed into other elements) and the resources are mapped to telemetry parameters. Actions are mapped to items on the FSW command schedule, and so on.

It is through this module that ISIS consumes other on-board PUS services: the On-Board Operations Scheduling, Event-Action, Housekeeping and Diagnostic Data Reporting and On-board Monitoring.

The Real-World Interface module keeps a real-time database that stores actual values read from sensors. As in any real-time database, the stored data has a timing constraint, which informs for how long a previously-read value is valid. If the database is queried for an 'old' value, it triggers a new sensor reading, stores the value and just then sends it to the model.

This module also has an events occurrence table, updated from the ground segment, that is needed to compute the effects of the predicted events.

## C. On-board Prediction of Potential Faults and Model Validation

ISIS, as any PUS service, can be monitored and controlled from ground through telecommands. It can also be consumed by an on-board application, which triggers inference sessions. An inference session starts 'now' (that is, from the current moment) and has an ending time that is defined by the last command on the schedule or the last event predicted on the events occurrence table (plus a delta-T). The results of the inference session are then made available for the caller application until another session is started.

To validate ISIS and its models, we are developing an on-board application that triggers inference sessions on each change that is made on the command schedule or on the events occurrence table.

With the results of the inference session, the application performs two main monitoring activities:

1. Checks regularly if the states predicted by the model are observed on the real system (there is an acceptable margin of difference for each parameter). If one of the predicted states is shown to be different from the actual state, ISIS enters in halt mode and sends a report to ground informing that the model has to be updated. In this case, the model can be wrong, or the satellite behavior may have changed (because of an equipment unexpected failure, for example).

2. Monitors the inferred parameters and sends reports to the ground when ISIS shows that a monitored parameter will be off-limits because of the last change on the operations plan.

## D. ISIS as an Autonomy Kernel

ISIS and its monitoring and model validation application allow us to reach our proposed autonomy level F3. ISIS is also intended to be an autonomy kernel, since applications for monitoring, replanning and

[†] We don't use the ECSS term 'activity' to refer to the space system monitoring and control functions. Instead, we have adopted the term 'action', to emphasize the lack of duration in our operators.

diagnosis/prognosis can perform queries and send commands to the service, such as:

- query if a given action is applicable (if it does not result on a violated constraint);
- individually submit actions and events to ISIS, and query what will be the resulting state;
- undo the last submitted action – it is not possible to undo events;
- query 'what are the applicable actions at a given state, at a given moment?';
- query 'what are the applicable actions that can change the state of a given system element at a given moment?';
- compare the states at two different given moments.

## VII. Final Remarks

ISIS is currently under development. Once the service is done, we will start exploring its use – along with the monitoring and model-validation application – using Lattes scientific satellite as a case study.

Together with the service, we are developing a modeling environment to help on the creation and maintenance of the SSM and the extraction and compilation of Domain-Specific Views from it.

ISIS integration with PUS services was pretty straightforward, since the services and its internal structures are ready to communicate both with the ground segment and with on-board applications. We use the services as they were first implemented – the only exception is the On-board Monitoring service, on which we had to create a new monitoring mode, 'inference'. In this mode, the reports sent to ground have a different treatment.

The alignment with the ECSS standards and its autonomy concepts and definitions allowed us to better organize our efforts to achieve a greater level of autonomy for INPE's satellites. The identified gap between the autonomy levels E3 and E4 was filled with a proposed intermediate level F3. This new level is being implemented by ISIS.

After a first validation on a real mission, ISIS should be ready to be an ECSS and PUS-compliant autonomy kernel for on-board replanning and diagnosis applications. Just then we will try to achieve the last autonomy level.

## References

[1]V. Verma, A. Jónsson, R. Simmons, T. Estlin and R.Levinson, "Survey of Command Execution Systems for NASA Spacecraft and Robots", in Plan Execution: A Reality Check Workshop at the International Conference on Automated Planning & Scheduling (ICAPS), 2005.

[2]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 (IAC), Turin, Italy, October 1997.

[3]ECSS. "Space Segment Operability", ECSS-E-ST-70-11C, ESA Requirements and Standards Division, July 2008.

[4]A. Jónsson, R. Morris and L. Pedersen, "Autonomy in Space Exploration: Current Capabilities and Future Challenges", in Proceedings of the IEEE Aerospace Conference, Big Sky, Montana, March 2007.

[5]ECSS. "Ground Systems and Operations – Telemetry and Telecommand Packet Utilization", ECSS-E-70-41A, ESA Publications Division, January 2003.

[6]M. Merri, B. Melton, S. Valera and A. Parkes, "The ECSS Packet Utilization Standard and its Support Tool", Proceedings of the 7th International Conference on Space Operations (SpaceOps '02), Houston, USA, October 2002.

[7]ECSS. "Ground Systems and Operations - Monitoring and Control Data Definition", ECSS-E-ST-70-31C, ESA Requirements and Standards Division, July 2008.

[8]ECSS. "Test and Operations Procedure Language", ECSS-E-ST-70-32C, ESA Requirements and Standards Division, July 2008.

[9]ECSS. "Spacecraft On-board Control Procedures", ECSS-E-ST-70-01C Draft 8.1, ESA Requirements and Standards Division, October 2008.

[10]D. Bernard, G. Dorais, E. Gamble, B. Kanefsky, J. Kurien, G. K. Man, W. Millar, N. Muscettola, P. Nayak, K. Rajan, N. Rouquette, B. Smith, W. Taylor and Y. W. Tung, "Spacecraft Autonomy Flight Experience: the DS-1 Remote Agent Experiment", in Proceedings of the AIAA 1999, Albuquerque, NM, USA, September 1999.

[11]S. Chien, R. Sherwood, D. Tran, R. Castaño, 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 Proceedings of the International Symposium on Artificial Intelligence Robotics and Automation in Space (I-SAIRAS), Nara, Japan, 2003.

[12]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.

[13]I.A. Nesnas, A. Wright, M. Bajracharya, R. Simmons, T. Estlin and W. S. Kim, "CLARAty: An Architecture for Reusable Robotic Software", in Proceedings of the SPIE Aerosense Conference, Orlando, Florida, April 2003.
[14]F. N. Kucinskis and M. G. V. Ferreira, "An On-board Knowledge Representation Tool for Satellite Autonomous Applications", in Proceedings of the 2008 ACM Symposium on Applied Computing, Fortaleza, Brazil, 2008.
[15]B. C. Williams and P. Nayak. "A Model-Based Approach to Reactive Self-Configuring Systems", in Proceedings of AAAI, 1996.
[16]S.C. Hayden, A.J. Sweet, S.E. Christa, D.Tran, S.Shulman. "Advanced Diagnostic System on Earth Observing One", in AIAA Space 2004 Conference and Exhibit, San Diego, USA, September 2004.
[17]S. Narasimhan1 and L. Brownston. "HyDE - A General Framework for Stochastic and Hybrid Model-based Diagnosis", in Proceedings of 18th International Workshop on Principles of Diagnosis (DX 07), Nashville, USA, May 2007.