# An AI Electrical Ground Support Equipment for Controlling and Testing a Space Instrument

María Dolores Rodríguez-Moreno, Manuel Prieto and Daniel Meziat
Departamento de Automática. Universidad de Alcalá
Ctra Madrid-Barcelona, Km. 33,6. 28871 Alcalá de Henares (Madrid), Spain.
{mdolores, mpm, meziat}@aut.uah.es

**Abstract**

An versatile and modular Electrical Ground Support Equipment (EGSE) system has been developed using Artificial Intelligence (AI) techniques to control and test the PESCA instrument and the communication process with the satellite without any human supervision. The PESCA instrument has been designed and built with the purpose of studying the Solar Energetic Particles and the Anomalous Cosmic Rays. It will be part of the Russian PHOTON satellite payload that is scheduled to launch in December of 2005.
The tool allows complete and autonomous control, verification and validation of the PESCA instrument, although its modularity makes it extensible to other on-boards instruments.

## 1. Introduction

Satellite domains are becoming a fashionable area of research within the Artificial Intelligence (AI) community due to the complexity of the problems that these domains need to solve. With the current USA and European focus on launching satellites for communication, broadcasting, scientific purpose or localization tasks, among others, the automatic control of these machines becomes an important problem. Of special attention in the last few years has been the development of an autonomous architecture that can carry out on the ground or on board, a large number of functions such as planning activities, tracking the spacecraft's internal hardware, and ensuring correct functioning and repair when possible, without (or little) human intervention. In these new models of operations, the scientists and engineers communicate high-level goals to the spacecraft, these goals are translated into planning and/or scheduling sequences; then a continuous check of the spacecraft status is verified in order to detect any damage, changes in the environment or in the goals; and finally execution is performed. It must also have the capability to understand the error/new situation that can occur during the process of accomplishing the goals.

Techniques from two different areas have been traditionally used: AI Planning and AI scheduling. Deliberative planners embody powerful techniques for reasoning about actions and their effects. They are good at finding precedences among activities but quite limited at resource or time reasoning. Scheduling systems are good at optimising and assigning time and resources to activities, but they require knowing order relations among the activities. A current approach to solve this problem, is to use a scheduler at the output of the planner in order to assign resources and time to each activity (this approach was used in the Deep Space One [5]).

Depending on the problem complexity some domains allow a strict separation between planning and scheduling. But, in other cases, there is an indirect temporal and resource dependency with other states and goals that cannot be taken into account if we separate both tasks. The above approach is weak if the scheduler fails to find a solution: expensive and unsatisfactory solutions can be again generated by the planner if it does not receive any feedback from the scheduler.

In this paper we present the integration of AI Planning and Scheduling Techniques into the EGSE software designed to allow autonomous control of the instrument PESCA that will be on-board the Russian PHOTON satellite for the acquisition of Energetic and Anomalous particles. It will also allow verifying and validating the instrument.

The paper is structured as follows: section 2 describes the related work to the one presented in this paper. Next, we present an overview of the PESCA instrument. Then, section 4 describes in detail the EGSE architecture with each module that it comprises. Finally conclusions and future work are outlined.

## 2. Related work

Several systems have faced the satellite domain problems such as DEVISER [18] that was used in the autonomous spacecraft Voyager which photographed Jupiter, Saturn and their satellites in 1979, 1980 and 1981. DEVISER adopts the style of a discrete event simulation system, in which all changes can be assumed to occur at specific points in time, rather than in continuous time simulations. For each activity, a duration and a start time window are presented.
SIPE [25] is a domain-independent planning that supports both automatic and interactive generation of hierarchical partially ordered plans. It was the first planner to handle consumable and producible resources. It assumes discrete time, states and operators.
Other more ambitious approaches have integrated planning and scheduling in the same system as HSTS [16] by instantiating state-variables into a temporal database. It has been applied to the problem of generating observation schedules for the Hubble Space Telescope. SPIKE [14] was initially used for science operations for the Hubble Space Telescope ground system, but then it was adapted to schedule a variety of astronomical scheduling problems.
The O-PLAN2 [23] integrates planning, scheduling, execution and monitoring. It is a domain independent, hierarchical tasks network, least commitment architecture that uses heuristic search strategy and keeps many partial plan solutions pawned at each decision point.
The New Millenium Remote Agent (NMRA) [15] was the first time an AI agent controlled for a six days period a NASA spacecraft: the Deep Space One (DS-1). NMRA is a hierarchical reactive system partially based on SAT-compilation that integrates real-time monitoring and control with constraints based-planning and scheduling, robust multi-threat execution and model-based diagnosis.
ASPEN [19] uses a rich representation in activities that easily allows the introduction of start times, end times, duration and use of one or more resources. It is part of CASPER [18] that allows working in dynamic environments as most of NASA projects require [4, 5, 6, 10, 26].
MAPGEN [1] is part of the Mars Exploration Rover mission. The planner system is called EUROPA [11] and it can be considered as an evolution of the Deep Space One planner [15].
CONSAT [20], an application that allows planning and scheduling nominal operations to perform in four satellites along the year for a commercial Spanish satellite company: HISPASAT.

## 3. The PESCA instrument

The PESCA instrument [8, 17] has been designed and built by the Space Research Group of the University of Alcalá with the purpose of studying the Solar Energetic Particles and the Anomalous Cosmic Rays. It will be part of the Russian PHOTON satellite payload that is scheduled to launch in December of 2005.

As depicted in figure 1, the PESCA instrument is composed of a telescope, made up of four silicon ion implanted detectors, which register the solar particle crossing through the detectors, the PIASE for signal amplification and analogue to digital conversion, and the PICAS, for the control of the whole instrument. PICAS is composed of a Data Processing Unit based on the Radiation Hardened MAS281 microprocessor and a set of interfaces with the rest of the instrument and the satellite. PICAS admits telecommands sent from ground for data acquisition and control parameter setting. The communication between ground and PICAS is performed through the Satellite On Board Data Handling (OBDH) computer. The communication process comprises two kinds of data, telecommands and telemetry. On the one hand, the telecommands are a set of tasks to be performed by the satellite or the payload. On the other hand, the telemetry is the response given by the satellite, which at the same time comprises the status data and the scientific data, which constitute the purpose of the mission.

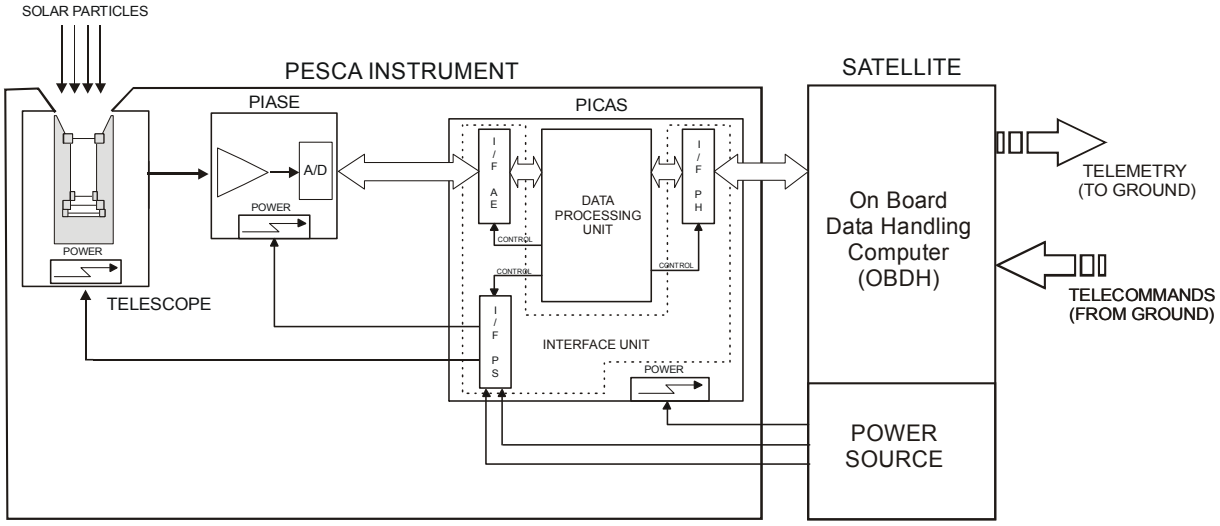


Figure 1: PESCA Instrument

To verify and validate the system in the laboratory, a complete set of simulators and software applications have been developed to perform the standardised test procedure:

- Unit tests. To check the correct functionality of each PESCA component.
- Integration tests. To check the correct interaction among each PESCA module.
- System tests. To check the correct functionality of the whole PESCA Instrument .
- Acceptance tests. To validate the user requirements. These final tests are carried out by the EGSE, that is commonly used in most of the space projects.

The test system that has been mounted in the laboratory to validate and verify the instrument is shown in figure 2. The main elements are: the PESCA instrument, an On Board Data Handling (OBDH) Emulator which fulfils the OBDH European Space Agency standard, a power source to supply the necessary voltages and currents to the PESCA instrument, and, finally, a PC computer with the Electrical Ground Support Equipment (EGSE) software.

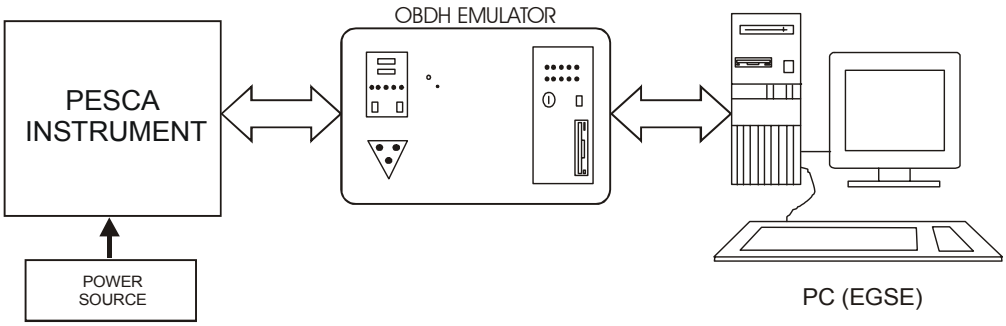


Figure 2: Test scenario.

The EGSE allows the complete control over the instrument and it has been a key tool during the whole acceptance test procedure. It basically allows the telecommand sending and data reception, thus making possible the verification and validation of the PESCA instrument. Since its first prototype, the EGSE has evolved to be adapted to the new needs that the whole system required. In the last version, the EGSE includes the use of AI techniques to allow autonomous control. The EGSE AI features will be described in the following sections.

## 4. The EGSE Architecture

The initial EGSE version was a simple application that only allowed sending telecommands and receiving raw data. The user interaction was through the command line, and the received data was stored in files.
The next version was developed in a graphical environment, and it included preprocessing capabilities that allow the on-line visualization of the received data, both scientific and housekeeping data, facilitating the data inspection. Then, due to the huge amount of possible test cases, we felt the need of executing those tests in autonomous mode, with low human supervision. The last version includes an integrated AI planner and scheduler system in order to schedule the test along the time.
The different versions have been developed in a modular way, in order to increase the functionally, without altering the basic operations. The EGSE is an object-oriented system that provides a reusable set of software components that makes it feasible to extend the control to more or different instruments.

The EGSE architecture is composed of the following modules, as figure 3 shows:

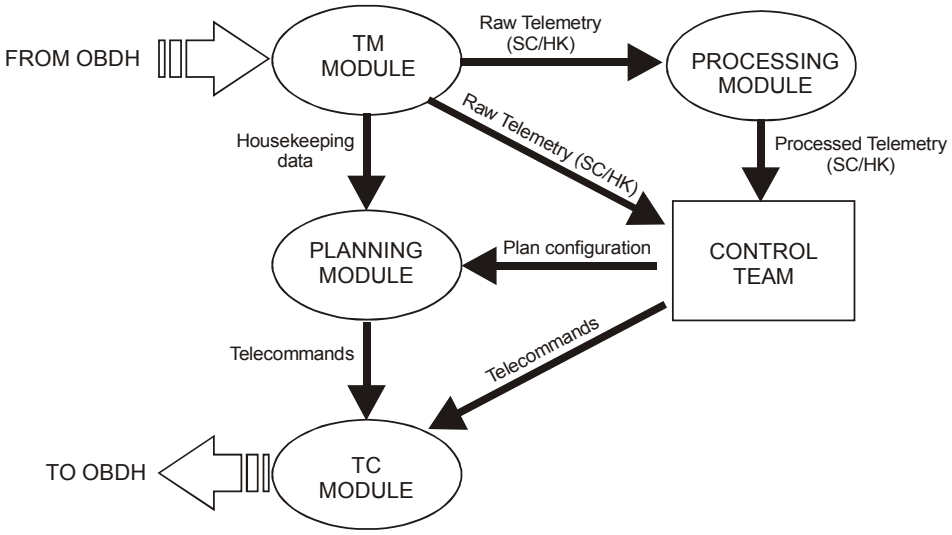


Figure 3: EGSE Architecture

The EGSE has the following features:

- Telecommand sending. The EGSE allows telecommand sending to both, the OBDH and the PESCA instrument. By means of these commands, the control team or the planning module can act over the PESCA instrument, changing the data acquisition parameters and other PESCA operation states such as detectors power on/off, coincidence/ anticoincidence mode setting, threshold setting, etc. Through the telecommand sending, the PESCA operation is tested.
- Telemetry reception. As mentioned before, the telemetry consists of two kinds of data: status data (housekeeping data) and scientific data. In the one hand, through the housekeeping data, the control team or the planning module knows the PESCA instrument status at any time (i.e. detector status) and therefore may act in consequence, for example in case of failure detection. In the other hand, scientific data contain the interest data, purpose of the instrument. That is, the energy lost in the detectors.
- Scientific Data processing. The user can make some processing over the received scientific data. This processing includes, among others, real time data visualization, data representation in different modes and data export to different formats.
- Planning/Scheduling. EGSE has an autonomous operation mode that avoids human supervision. In this mode, EGSE continuously checks housekeeping data and thanks to the integration of the planner IPSS [14] takes the specific actions to react against possible on flight failures.

The next subsections describe in detail the four main modules that composed the EGSE.

## 4.1. The Telecommand Module

This module allows, both the planner and the control team, to send the telecommands to the OBDH emulator or to the instrument itself. The communication with the OBDH emulator is performed through two RS232 serial ports completely configurable from the EGSE main window. In manual mode, the telecommands can be sent though a command line provided by the graphical interface or through a list control that displays the telecommands listed in a user defined file. This ASCII file, easily editable from a common text editor, contains the telecommand identifiers and the telecommand data bytes (header, body and terminator) to be sent through the serial port. The default file is the one for the PESCA instrument but the telecommand list can be modified through the interface. This gives versatility to the EGSE because it is not restricted to a fixed set of telecommands of a given telecommand format, and therefore, it is not restricted to a specific instrument. For every telecommand sent, the OBDH response is shown in order to know if the telecommand has been successfully delivered. The EGSE allows programming the telecommand sending in time (Time Tag Telecommands). From a menu option, the user can program a list of telecommands to be sent in a specific time and date. This feature allows setting a telecommand sequence for delivery. All telecommands issued and the OBDH responses are registered in a log file for later inspection. Log files are stored in ASCII format, legible from any text editor. Figure 4 shows an example of the telecommand interface window.

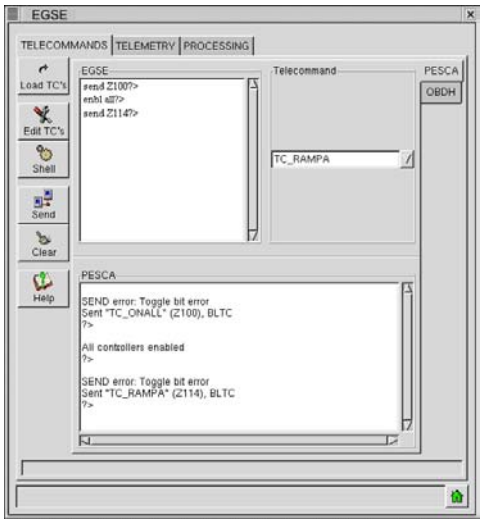


Figure 4. Telecommand interface window

In autonomous mode, the Telecommand Module checks if the planner has generated a new telecommand sequence file. This file contains the telecommands that the planner has determined necessary to send to the OBDH or to the Instrument to reach a specific solution. In this case, the telecommands are directly issued through the OBDH telecommand channel.

## 4.2. The Telemetry Module

This module receives the telemetry provided by the satellite OBDH and distributes it to the control team or to the planner. The telemetry data are received according to the loaded telemetry profile and are divided into housekeeping data and scientific data. Both data types are stored in different files in raw binary format. In autonomous operation, these files are the input to the Planner. For manual operations, the EGSE provides a graphical interface through which the controller can configure some parameters related with the telemetry acquisition (such as the number of data packets to collect) as well as control the acquisition process. In both cases, the telemetry profiles are user defined, allowing the EGSE configuration for other instruments. The EGSE permits both programmed acquisition and continuous acquisition, handling automatically the corresponding data files.

Figure 5, shows a typical scientific data acquisition from the PESCA instrument in manual mode, where detector 2 data are represented versus detector 3 data. In this case, the results are represented in energy channels, where each channel corresponds to a specific particle energy.

As it can be seen in the figure, there are three possible options. The *Start* button activates the data acquisition according to several parameters (i.e. date, time, number of packets). The *Stop* button halts the data acquisition, and finally, the *Analyser* button launches a complete protocol analyser for direct data inspection.

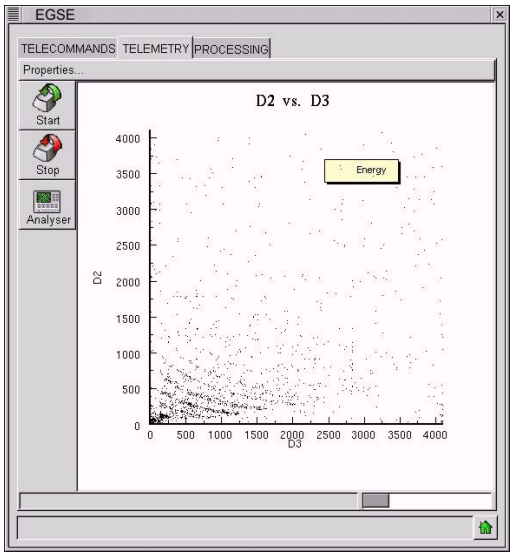


Figure 5. Telemetry interface window

## *4.3. The Data Processing Module*

After the data acquisition, both housekeeping data and scientific data may be displayed and processed through the data processing option. Scientific data may be displayed in a histogram form, transforming the EGSE in a multichannel analyser, or representing the data in two axes. As stated in the previous section, all the data received are stored in binary format. In order to allow the data export to other data processing programs, the EGSE allows the data conversion into the ASCII format. Figure 6 shows an example of data displaying in multichannel mode. It basically shows the count rate of each registered particle. In this example, the D1 histogram is displayed in red colour, whereas D3 histogram is represented in yellow.

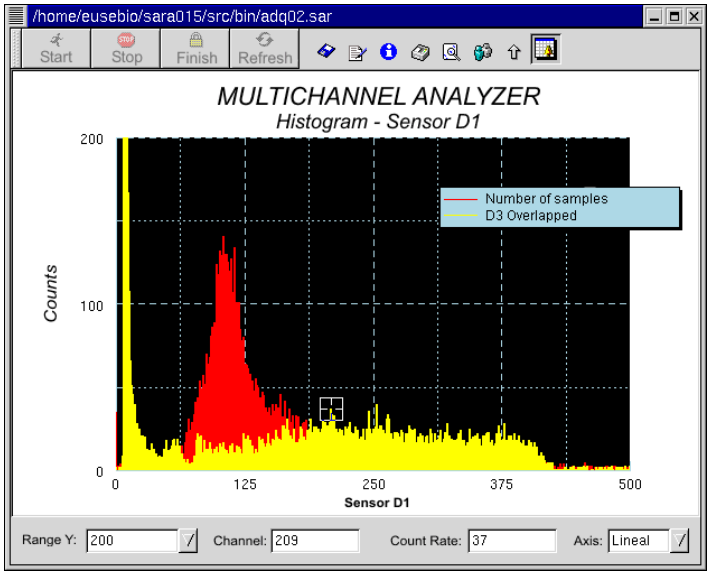


Figure 6. Example of processing window

## 4.4. The Planning Module

The EGSE program also has the capability of working in an autonomous mode thanks to the integration of AI planning and scheduling techniques. We have used the Integrated Planning and Scheduling System (IPSS) [21] for this task but any other planner can be used (i.e. LPG [12], O-PLAN-2 [23] or IxTeT [13]). Although IPSS presents some features that make it feasible to use in this domain such as minimising the makespan or establish different quality metrics.
The EGSE architecture makes the integration possible because it is based on a file input/out system. The telemetry file generated by the telemetry module, is translated into the specific planner syntax (PDL 4.0 [3] in our case or it can be also adapted to the PDDL 2.2 standard [9]). As depicted in figure 7, the Planning Module takes this telemetry and generates a problem file with this information. This file joined to the already defined domain and control knowledge file allows running the planner (see section 3.4.2 to know more about the inputs and outputs of IPSS). The planner output is saved into another file that will be manipulated to generate the input format to the EGSE.

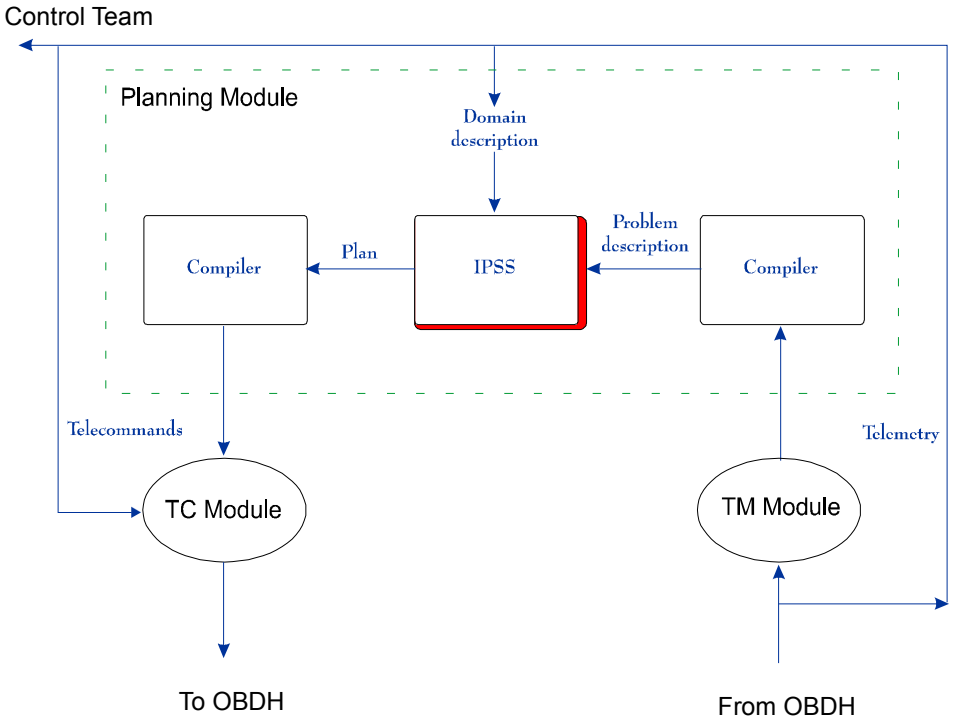


Figure 7: Inputs/Outputs of the planning module

### 4.4.1. IPSS: An integrated system

In IPSS [21] the reasoning is subdivided in two levels. The planner focuses on the actions selection than time-resource usage, and the scheduler (a CSP approach) on the time and resource assignments. The basic problem of this kind of integration is to choose a modality to exchange data that preserves the efforts of each component and puts the other components in conditions to contribute positively to the reasoning.

Figure 8 shows the Planning Module architecture with the four layers that compose IPSS. We have subdivided IPSS into two groups: IPSS-P that corresponds to the planning reasoner, and IPSS-S that corresponds to the scheduling reasoner. We can also see the two inputs to the Planning Module (Domain Description and Problem Description) and the output generated (that is, the plan).

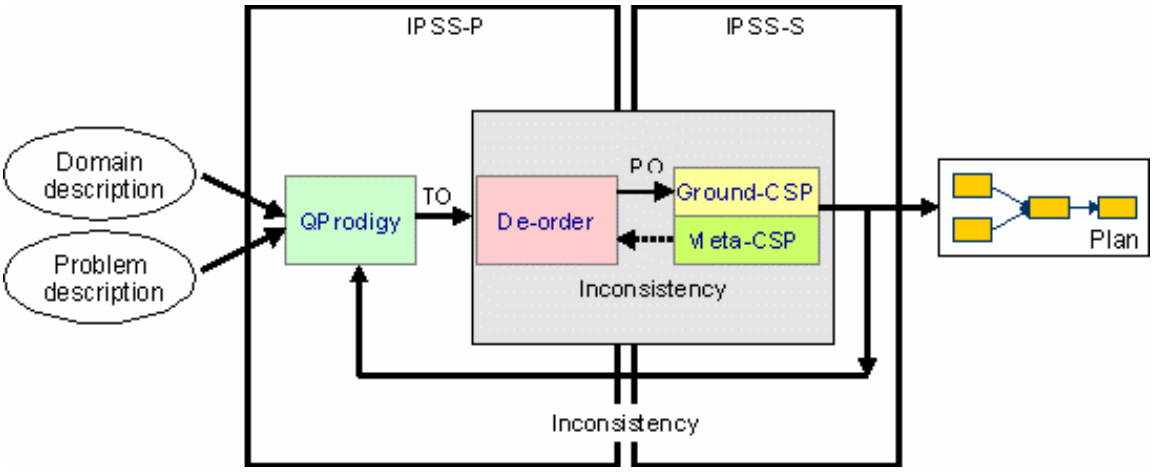


Figure 8: Planning Module: IPSS architecture

The Plan Reasoner (IPSS-P) is composed of the heuristic planner QPRODIGY [2] and the Deorder algorithm. The planner performs a bi-directional search: it begins by performing backward search from the goals, selecting which goals to achieve, which operator to use to achieve the corresponding goal, and which bindings to use for its variables. As soon as it can apply any selected operator, it decides whether applying it or continue subgoaling. If, at any decision point, it applies one operator, the planner consults IPSS-R for the time and resource consistency. If the resource-time reasoner finds the plan inconsistent, then the planner backtracks. If not, the operator gets applied, the current state is modified, and the search continues.
The planner we are using generates total ordered (TO) plans. This type of plans is not appropriate if we are looking for time and resource optimisation. That is, some orderings can be removed from the incomplete plan being generated by the planner, allowing parallel executions of operators. This is referred as deordering of the solution, and IPSS-P has a module that transforms the incomplete to plan into an incomplete PO plan.

The Scheduler reasoner, IPSS-S, is represented as a Constraint Satisfaction Problem (CSP) partitioned in two sub-problems. A basic Ground-CSP to reason on temporal constraints and a Meta-CSP to reason on resource constraints.

The Ground-CSP layer represents the set of temporal constraints as a Simple Temporal Problem [7]. In particular, significant events, as start/end time of operators are represented as temporal variable $tp_i$ called time points. And each temporal constraint has the form $a \leq tp_i - tp_j \leq b$, where $tp_i - tp_j$ are time points and a and b are constants. For example, let $tp_s$ and $tp_e$ be respectively the start and end time of an operator p, its duration constraints can be represented as $d \leq tp_e - tp_s \leq d$, where d is its duration.
For solving the Meta-CSP sub-problem we use the algorithm for reasoning on binary capacity resources proposed in [22]. Under this representation model, resource conflicts are computable in polynomial time because conflicts are represented as pairs of temporally overlapping activities that require the same resource.
The algorithm iteratively imposes ordering constraints for solving resource conflicts between activities that require the same resource. When there is more than one possible order it chooses as heuristic the link following the planner logical order. Again, in case of an unsolvable resource conflict (no precedence posting can solve the contention) a failure for resource inconsistency is sent to the planner that backtracks the last decision.

### 4.4.2. IPSS inputs and output

IPSS has three inputs: the domain theory, that is based on a STRIPS extension, the problem and the control rules or control knowledge that guides the problem solver avoiding backtracking.

The first step to follow to define the domain theory in the PESCA problem is to define variables. They are handled within the operator. For that, one needs to declare the type of each variable. The type hierarchy in IPSS forms a tree structure. Each type in the hierarchy belongs to a super-type called root and each type may have several subtypes. All the components that are part of the control system are represented as a sub-type of

the super-type. For example, PICAS, OBDH, LOBT (Local On Board Time), etc are sub-types of the root and this is represented as shown in figure 9.

```
(ptype-of DETECTOR :top-type)
(ptype-of LOBT :top-type)
(ptype-of PIASE :top-type)
```

Figure 9: Types in IPSS.

Some types have an infinite number of instances (such as threshold setting), they are called infinite types. This type of variables are generated when IPSS needs to instantiate the corresponding operator.
Figure 10 shows the syntax of a IPSS operator to turn on a detector. With respect to the pre-conditions, this operator has two: the status of the detector, o., and the status of the PIASE, on. As effects, the detector will be on.

```
(Operator OnDetector
  (preconds
    ((<dect> DETECTOR)
     (<Eanal> PIASE))
       (and (statis <dect> OFF)
            (on <Eanal>)))
  (effects
   ()
    ((del(statis <dect> OFF))
    (add(statis <dect> ON)))))
```

Figure 10: IPSS Operator to switch on a detector.

In some cases one telecommand correspond to one IPSS operator but in others, several telecommands are translated into one IPSS operator. The user can select from the interface what telecommand(s) correspond to each IPSS operator.

The second input to the planner is the problem, described in terms of an initial state and goals. Figure 11 shows a small fraction of the initial conditions and goals. The initial conditions are automatically generated from the telemetry reception process, that is, all knowledge that EGSE has about PESCA. The goal(s) are chosen from a list of goals available through the interface. In the example, one wants that PESCA acquire scientific data from any of the possible modes that the detectors can be, without any interruption. In PESCA there are defined four modes, in each mode a detector is not operative.

```
(state
  (and
      (on OBDH)
      (no-synchronize lobt)
      (off CDPUNOM)
      (off PIASE)
      (statis d1 OFF)
      (statis d2 OFF)
      (statis d3 OFF)
      (statis d4 OFF)
      ... ))
 (goal (scientific-acquisition data))
```

Figure 11: Some initial conditions and goals for the problem of acquiring scientific data.

When there is more than one decision to be made at decision points, the third input to the planner, the control knowledge (declaratively expressed as control rules) could guide the problem solver to the correct branch of the search tree avoiding backtracking. There are three types of rules: selection, preference or rejection. One can use them to choose an operator, a binding, a goal, or deciding whether to apply an operator or continue sub-goaling. Figure 12 shows an example of a selection operator rule. It says that if the PIASE is on and the

detector 2 is o. or broken then chose the ChangeMode2 operator. This control rule prunes the search tree and chooses the correct mode to detector 2.

```
(Control-Rule select-op-ChangeMode2
    (if (and (current-goal (statis <piase> ON))
             (or (true-in-state (statis d2 OFF))
                 (true-in-state (statis d2 BROKEN)))
             (type-of-object d2 DETECTOR)))
    (then select operator ChangeMode2))
```

Figure 12: Control rule to select the detector Mode.

As a result, IPSS generates a plan with the sequence of operators/telecommands that achieve a state (from the initial state) that satisfies the goal(s). The obtained plan does not consider any optimisation with respect to resource use or availability, given that for PESCA it is enough to find a plan. However, the planner could obtain an optimal plan according to some criteria that can be defined in the operators.

### 4.4.3. IPSS global behaviour

Figure 13 shows the planning main window. In the left window we can see the status of the different PESCA components through icons. The right window shows the same results written. The central window appears when the planning module is connected by the first time to the OBDH.
Every time a telecommand is sent following the plan generated by IPSS, an exhaustive check of the operator effects is performed in order to detect differences between the operator effects and the status data received after the telecommands sent. If any difference is found, a re-planning process starts. If a solution cannot be obtained, the program will abort and a log file will be generated with all the decisions made by the planner.
This information is translated into a suitable format for input to the IPSS (the problem file). This file joined to the already defined domain and control knowledge files allows running the planner. The IPSS output is saved into another ASCII file that will be manipulated to generate the telecommands needed to solve the problem.

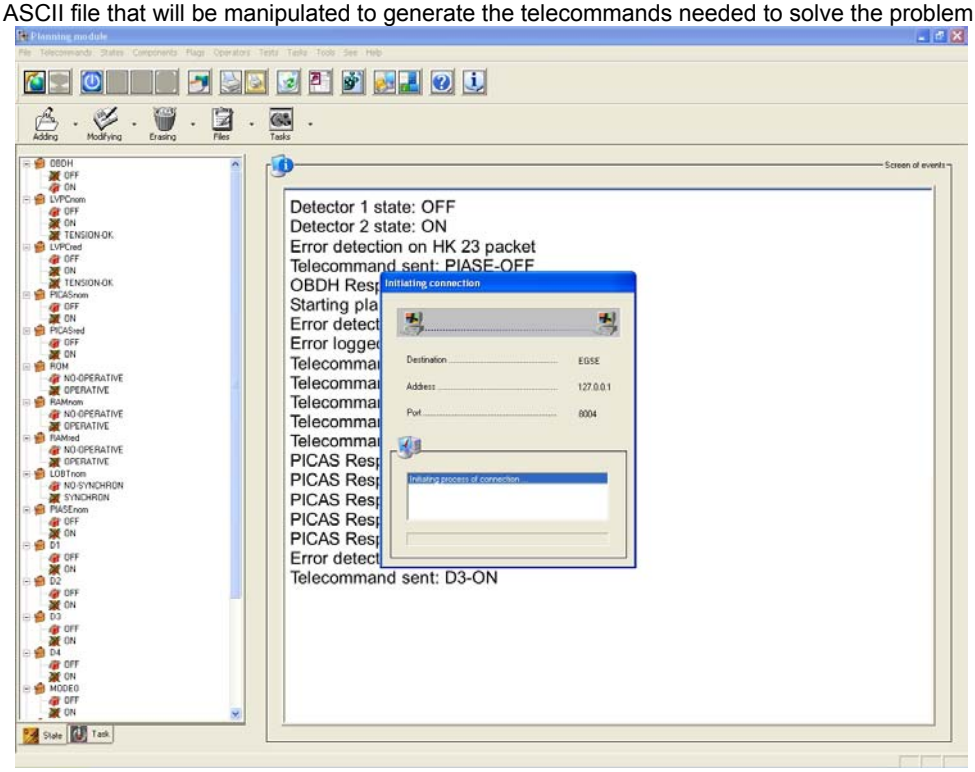


Figure 13: Planning module main window.

For example, one of the possible actions in the plan if one wants to receive scientific data will be to turn on the detectors. If in this process, the program realises that the detector is still off, the planner will generate a new plan. In this new plan, one of the actions will be to send the same telecommand three more times (sometimes the problem is not due to a detector damage but a bad transmission/reception).
If this fails, a new plan is generated to find a different mode of acquisition but if no valid solution can be obtained, the program will abort and a log file will be generated with dates and decisions made by the planner. It is also possible, when EGSE is running in auto mode, that the user interacts or inhibits the planner decision. This module has also let us to automatically perform the unit test, integration test and system test that PESCA needs before the PHOTON integration. All the tests need to be performed several times, varying the types of particles, the detector mode, the target and the redundant components.

Thanks to the planning process we have successfully run the tests with the minimum human cost. Figure 14 shows the main to window that allows temporising at what date and the different tasks that one wish to perform. In this case, we want to start the 7th September at 14:38. Among the tests that we want to perform are: to turn on the OBDH, synchronise the LOBT and turn on the detectors one and two.

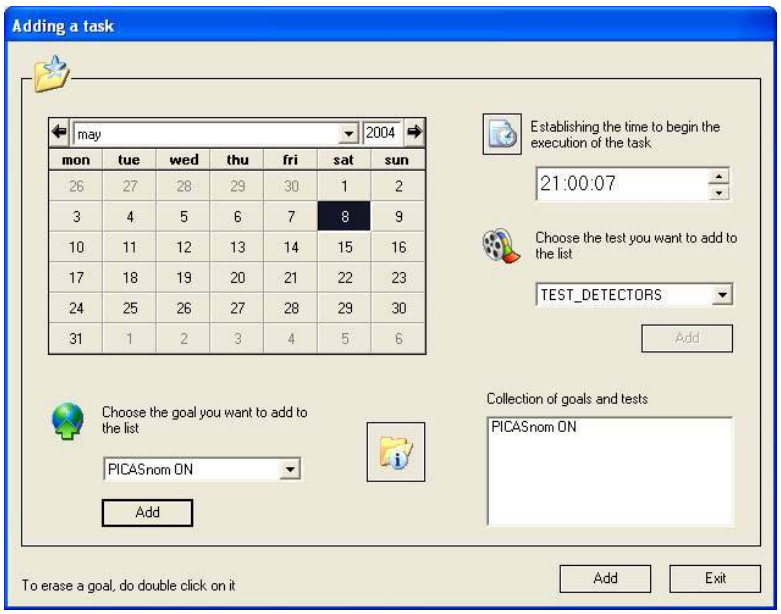


Figure 8: Main window to temporise tests.

## 5. Conclusions

We have presented in this paper the EGSE system for the control and test of the PESCA instrument that will be part of the Russian PHOTON satellite payload. The software has been successfully proved in the laboratory, and it has become a key part in PESCA testing. One of the main modules of the EGSE is the planning module. As it has been shown through the paper, the use of an AI integrated planning and scheduling system allows automating all the tests required for the PESCA instrument. These tests are essential for the integration with PHOTON satellite. It has been tested its efficiency during the develop of the instrument calibration tests in Ganil (France).
This EGSE has a great versatility, and it may be extended in the future to other instruments modifying several parameters thanks to its modular design.

## Acknowledgements

This work has been supported by Spanish CICYT (grant ESP99-1066-C02), by the European Community-Access to Research Infrastructure action of the Improving Human Potential Programme, No HPRICT1999-00019, by the NATO grant PST.CLG.977003 and partially funded by the CICYT project TAP1999-0535-C02-02. We also want to thanks Daniel Borrajo for the QPRODIGY support and Javier Andrés Alonso for the implementation in the planning module.

## References


[1] Ai-Chang, M., Bresina, J., Charesty, L., Hsu, J., Jónsson, A., Kanefsky, B., Maldaguey, P., Morris, P., and Rajan, K. Yglesias, J. *MAPGEN Planner: Mixed-Initiative Activity Planning for the Mars Exploration Rover Mission.* In Procs. of the 7th International Symposium on AI, Robotics and Automation in Space (i-SAIRAS) (2003).

[2] Borrajo, D., Vegas, S., and Veloso, M. *Quality-Based Learning for Planning*. In Working notes of the IJCAI'01 Workshop on Planning with Resources. (2001).

[3] Carbonell, J. G., Blythe, J., Etzioni, O., Gil, Y., Joseph, R., Kahn, D., Knoblock, C., Minton, S., Pérez, A., Reilly, S., Veloso, M., and Wang, X. *PRODIGY4.0: The Manual and Tutorial*. Tech. Rep. CMU-CS-92-150, Department of Computer Science, 1992.

[4] Chien, S., Rabideau, G., Willis, J., and Mann, T. *Automating Planning and Scheduling of Shuttle Payload Operations*. Artificial Intelligence Journal 114 (1999), 239–255.

[5] Chien, S., Rabideau, G., Willis, J., and Mann, T. *RADARSAT-MAMM Automated Mission Planner*. AI Magazine 23, 2 (2002), 25–36.

[6] Chien, S., Sherwood, R., Tran, D., Castano, R., Cichy, B., Davies, A., Rabideau, G., Tang, N., Burl, M., Mandl, D., Frye, S., Hengemihle, J. DAgostino, 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 Procs. of the 7th International Symposium on AI, Robotics and Automation in Space (i-SAIRAS) (2003).

[7] Dechter, R., Meiri, I., and Pearl, J. *Temporal Constraint Networks*. Artificial Intelligence 49 (1991), 61–95.

[8] del Peral, L., Bronchalo, E., Medina, J., Rodríguez-Frías, M. D., Sánchez, S., and Meziat, D. *An electronic device that suits space research requirements for ion detection*. IEEE Transaction on Nuclear Science 44 (1997), 1442–1447.

[9] Edelkamp, S., and Hoffmann, J. *PDDL2.2: The Language for the Classical Part of the 4th International Planning Competition*. Tech. Rep. Technical Report No. 195, 2004.

[10] Estlin, T., Rabideau, G., Mutz, D., and Chien, S*. Using Continuous Planning Techniques to Coordinate Multiple Rovers*. In Procs. of the IJCAI99 Workshop on Scheduling and Planning meet Real-time Monitoring in a Dynamic and Uncertain World. (1999).

[11] Frank, J., Jónsson, A., and Morris, P. *On Reformulating Planning as Dynamic Constraint Satisfaction*. In Procs. of the Symposium on Abstraction, Reformulation and Aproximation (SARA). (2000).

[12] Gerevini, A., Saetti, A., and Serina, I. *Planning through Stochastic Local Search and Temporal Action Graphs*. Jounal of Artificial Intelligence Research 20 (2003), 239–290.

[13] Ghallab, M., and Laruelle, H. *Representation and control in IxTeT, a temporal planner*. In Procs. Of the second international conference on AI planning systems (AIPS-94) (1994).

[14] Johnston, M. D. *SPIKE: Intelligent Scheduling of Hubble Space Telescope Observations.* Intelligent Scheduling. Morgan Kaufman, 1994, pp. 391–422.

[15] Muscettola, N., and Smith, B. On-Board Planning for New Millenium Deep Space One Autonomy. In *Procs. of IEEE Aerospace Conference, Snowmass, CO.* (1997).

[16] Muscettola, N., Smith, S., Cesta, A., and D'Aloisi, D. *Coordinating Space Telescope Operations in an Integrated Planning and Scheduling Architecture*. IEEE Control Systems 12 (1992), 28–37.

[17] Prieto, M., Martín, C., Quesada, M., Meziat, D., Medina, J., Sánchez, S., and Rodríguez-Frías, M. D. *Control and acquisition system of a space instrument for cosmic ray measurement*. In Nuclear Instruments and Methods in Physics Research, A443 (1999), pp. 264–276.

[18] Rabideau, G., Chien, S., Knight, R., Sherwood, R., Engelhardt, B., Mutz, D., Estlin, T., Smith, B., Fisher, F., Barrett, T., Stebbins, G., and Tran, D. *ASPEN - Automating Space Mission Operations using Automated Planning and Scheduling*. In SpaceOps 2000, Toulouse, France (2000).

[19] Rabideau, G., Knight, R., Chien, S., Fukunaga, A., and Govindjee, A. *Iterative Repair Planning for Spacecraft Operations in the aspen System*. In Procs. of the 5th International

Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS), Noordwijk, The Netherlands. (1999).

[20] R-Moreno, M. D., Borrajo, D., and Meziat, D. *An AI Planning-based Tool for Scheduling Satellite Nominal Operations*. AI Magazine, to appear (2004).

[21] R-Moreno, M. D., Oddi, A., Borrajo, D., Cesta, A., and Meziat, D. *IPSS: Integrating Hybrid Reasoners for Planning and Scheduling*. In Procs. of the 16th European Conference on Artificial Intelligence, ECAI04 (2004).

[22] Smith, S., and Cheng, C. *Slack-Based Heuristics for Constraint Satisfaction Scheduling*. In Procs. of the 11th National Conference on AI (AAAI-93) (1993).

[23] Tate, A., Drabble, B., and R., K. *O-Plan2: An Open Architecture for Command, Planning, and Control.* Morgan Kaufman, 1994.

[24] Vere, S. A. Planning in Time: *Windows and Durations for Activities and Goals*. IEEE Transactions on Pattern Analysis and Machine Intelligence pami-5 (1983).

[25] Wilkins, D. E. *Practical Planning: Extending the Classical AI Planning Paradigm.* Morgan Kaufman, 1988.

[26] Willis, J., Rabideau, G., and Wilklow, C. *The Citizen Explorer Scheduling System*. In *Procs.of the IEEE Aerospace Conference.* (1999).