A FIRST APPROACH FOR THE AUTONOMY OF THE EXOMARS ROVER USING A 3-TIER ARCHITECTURE

Pablo Muñoz, María D. R-Moreno, and Agustin Martínez

Departmento de Automática, Universidad de Alcalá, 28871 Alcalá de Henares, Spain
Email: {pmunoz, mdolores, hellin}@aut.uah.es

ABSTRACT

The ExoMars rover of the ESA-NASA programme aims to be the first robot to drill the Martian surface. Given the complexity of the operations to follow and the conditions of the mission (distance, delay, security, etc.), we present a first approach to make an autonomous system to manage the rover using a 3-Tier architecture. This system is specially focused on the design and description of the models that represent the rover subsystems and the terrain data. The skills that we want to include in the system are: images acquisition or drilling at particular locations and be able to follow a predefined path.

1. INTRODUCTION

Mars exploration is a trend topic in the near future for both NASA and ESA. An important part of the investigation is the surface exploration and in situ experiments. To do this, it is scheduled the launch of NASA Mars Science Laboratory in November 2011 and next, in 2018 it is intended to launch ESA-NASA ExoMars. The First has the most advanced suite of instruments for scientific studies ever sent to the martian surface, and the second wants to be the first to investigate the subsurface of Mars using an advanced drill equipment.

The operation of a planetary rover is a complex task, due to the hard environment conditions and the communications problems (i.e. delay between ground operation team and communication windows). Therefore, it is interesting to try to automate the rover operations, essentially the critical rover subsystems. To bring out that question, in this paper we propose a modular 3-Tier or 3T architecture in which each layer has a different technology.

The employed architecture was initially designed to automate the locomotion of the Ptinto exploration robot [1]. In this version, the robot models have been replaced by new models adapted to the ExoMars capabilities, and the functional layer uses GENerator Of Modules ($G^{en}oM$) [2], made by the LAAS-CNRS. The execution layer continues using PLan EXecution Interchange Language (PLEXIL) and Plexil Executive [3] from NASA and Carnegie Mellon University. For the high tier we still use the Planning Domain Definition Language (PDDL) [4] to model the tasks and $SGplan_6$ [5] as the planner to generate the solution, and our own library to manage the planner and the data contained in the problem file.

For test and validation of the architecture we have used the ExoMars rover model in the 3DROV and Simsat ESA simulation tool [6]. The couple 3DROV and Simsat is an advanced planetary rover system simulation environment that includes models of the planetary environment, the power, the mechanical and the scientific subsystem of the rover, a generic onboard controller and a ground control station module. Therefore, the functional layer is designed to work with the generic controller of 3DROV.

The paper is structured as follows. First, a brief review of the tools used in this project is provided. Next, section 3 presents our proposed design to control ExoMars rover. Finally, some conclusions and future work are presented.

2. TOOLS USED

In this section we present a brief review of the tools employed in this work. First the language and the planner used for the deliberator layer is presented. Next a short description of the executive is provided. Finally, the tools used for implementing the functional layer and the simulation environment are described.

2.1. PDDL language and planner

There are several kinds of planners, some of them are domain-dependent and others are domain-independent (that is, for generic purposes) ones. In this case we have chosen a general purpose planner that uses the Planning Domain Definition Language (PDDL) [4]. The reason to select a general purpose planner with a standard language is the possibility of replacing the planner for another with low cost. The same domain/problem files can be used without (or with minimal) modifications, and, if the planner support new PDDL features or extensions (like PIPSS [7], that implements resources as a PDDL extension), we

will be able to extend the possibilities and semantics of the deliberator system in a short period of time.

PDDL planners are systems that use two input files to represent their knowledge base. One of the files contains a description of the actions that represent "what it can/(cannot) be done". And the other file includes the three elements which define the problem: the objects of the world, the initial state and the goals we want to achieve. These elements allow us to model the rover subsystems, the terrain in which the rover is and the desired science objectives. With this information, the planner searches a sequence of actions that can reach the goals from the initial state. The optimal solution of the problem is a conjunction of two factors: the metric used and the resolution algorithm. For our first version of the ExoMars architecture we continue using SGPlan version 6 [5] as the planner.

2.2. PLEXIL language and executive

For the executor we also maintain the couple PLEXIL [3] and its associated executor, the Plexil Executive. PLEXIL is a structured language to make flexible and reliable command execution plans. It is portable, lightweight, deterministic (given the same sequence of events from the external world) and very expressive. Plexil Executive is the interpreter, designed to facilitate the inter-operability of execution and planning frameworks. It has an interface module to connect the executor with one or more external system.

A PLEXIL plan is a tree of nodes which represents a hierarchical breakdown of tasks. Each node has a set of conditions to control its execution and a section that describes its function. With all these elements we can express *if-then* branches, loops and batch processes and use subplans that would be executed only when a special situation occurs.

Plexil Executive can be divided into three modules. First, the executive core that implements the PLEXIL syntax and the algorithms to execute the plans written in PLEXIL. Then, there is a module for the management of the resources in case there were conflicts when they were used (it is a basic scheduler called Resource Arbiter). Finally, there is an external interface system to connect the executor with an external system. The interface must be implemented by the user for each external system that we want to control from the executor level.

2.3. G^{en}oM framework

The Generator Of Modules (G^{en}oM) [2] is a framework that allows the definition of standardized modules. A module consists of a formal definition of the internal structures and the services that provides, and the associated software code which is responsible of the services execution. The framework allows us to model a subsystem into one (or more) module(s) that specifies the internal data of the module (status, associated data of the subsystem, etc.), possibly an external information to share with other modules or external systems (called *posters*), and, finally, a set of requests that perform the functionality of the module.

The creation of a module consists of two steps: a file definition (data structures, requests interface, etc.) and services implementation of the requests into a piece of software called *codels* (which stands for code elements). When these elements are already generated (both can be refined a posteriori) the automated tools of the framework do the rest. The result is a server, which integrates the services and data of the module, and a serie of interface libraries to access to the *posters* and their requests. A G^{en}oM module provides a portable and reliable service, and lets us to integrate initialization services, interrupt routines, handle of failures, parameters checking and coordination of services. It is interest to highlight that the module works in a server-client paradigm, that allow to connect more than one client to the module, which could be useful in order to monitor the module in real-time data or enable access to human controlled interfaces.

2.4. 3DROV Simulation Environment

The 3DROV planetary rover simulation environment [6] allows early-stage virtual modelling of terrain and mobile robots systems. The simulation environment is composed of multiple modules connected through standardized interfaces:

- The *Simulation Framework*, that is the ESA's Simsat, responsible of the execution and scheduling of the simulation.
- The *Control Station* which provides a virtual ground control station.
- The *Generic Controller* that manages the onboard flight software. This module allows us to connect software modules to control the rover.
- *Rover subsystems models*, including the models of rover physical subsystems, sensors and scientific instruments.
- *Environment*, a block in charge of the timekeeping and terrain and atmospheric conditions.
- The *Visualization Environment* (shown in Fig. 1), a front-end that provides real-time visualization of the simulation progress.

We use a 3DROV version that includes a large terrain model of the Mars surface and the ExoMars model with the power, locomotion, drill, cameras and mast subsystems.

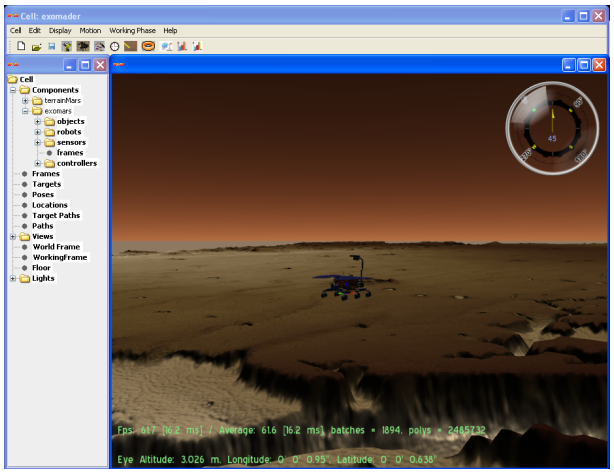

Figure 1. 3DROV visualization tool.

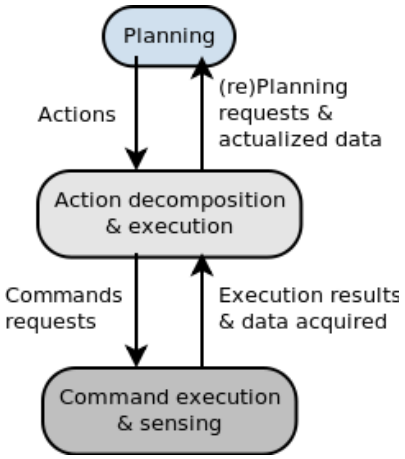

Figure 2. Execution flow.

3. THE PROPOSED ARCHITECTURE

The proposed architecture for the control of the ExoMars rover corresponds to a hybrid 3T-layer, in which the deliberator is the responsible of the plan generation for safely reach the designed scientific objectives and the storage of the long-term memory. We use a planning system based on PDDL with a specific domain and problem that describes the rover parameters, abilities and environment. This information is periodically updated with the information collected from lower layers. The middle layer or execution is in charge of the actions execution and synchronization. In this layer we use PLEXIL, a language of high level actions, and Plexil Executive to run the plans modelled in PLEXIL. This tier is a set of PLEXIL plans that handle the correct execution of the sequence activities reached by the deliberator. There are some PLEXIL plans to manage the different rover subsystems, to control the planning process and to activate fail-safe modes. Finally, the functional layer is associated to the hardware controller and must be able to receive commands requests and aborts, and relay the rover status and information collected by the sensors to the executor. The functional layer uses G^{en}oM, so each G^{en}oM module has the control of a subsystem that integrates both the behaviour of the subsystem and the interconnections between the other modules. Also, the functional layer and the executor have a series of rules that trigger reactive behaviours in order to response in a short time to eventual situations that may occur in both, the environment or in the internal state of the robot.

The execution flow of the system is shown in Fig. 2. First, the planner must obtain a valid plan that satisfies the higher number of science targets that are set. This plan must be read by the executor, who takes each action and decomposes it in a group of lower level commands. This layer can modify the execution order of the actions, for example, when a group of actions can be executed in parallel. Finally, the functional layer executes the commands sent by the executor and relays the execution result and other data (such internal state of subsystems or science data) to the executor. The executor is in charge of analyzing the results and act accordingly. When all are correct, the execution continues normally, but if there is any failure, the executor must take into consideration the kind of failure in order to entering into a safety state, or, more typically, detect a problem in the plan execution (an unknown obstacle for example). In that case, the executor needs to update the information of the problem as a previously step to call replanning at the high layer.

Figure 3 shows a conceptual view of the architecture. Next subsections aim to briefly detail the models of the architecture, from the functional layer to the deliberator.

3.1. Functional layer

The functional layer is the responsible of the management of the rover subsystems. Since the architecture works with the ExoMars model of the 3DROV, the hardware to control is the Generic Controller of the simulator suite. The identified subsystems of the ExoMars rover are: power, locomotion, drill, cameras, mast and communications. Due to the complexity of each subsystem, some of these are not yet implemented, but it is planned to integrate them in the future.

As mentioned above, the functional layer is implemented with G^{en}oM modules. Each module has a serie of requests that represent the activities that the subsystem can do, and a set of data that could be exported via *posters* to other modules or upper layers. Next there is a brief description of each module:

- *Power*: this module implements the battery model of the rover and its related parameters, that is, voltage levels, maximum instant consumption and the list of active subsystems. It periodically updates the battery status and lets the rest of modules to power on and off its power line. When a subsystem wants to

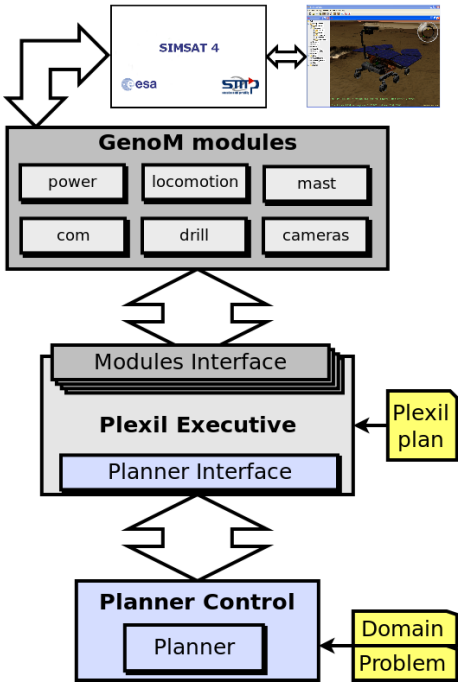

Figure 3. Architecture concept.

power on, the Power module can deny the request if the power consumption exceeds the limits, or when the battery is low and the subsystem to power on is not critical. It is important to note that all subsystems must power on their energy line and actualize their energy consumption before performing any operation. The Power module can also increment the battery level from the energy provided by the solar panels.

- *Locomotion*: the Locomotion module is in charge of both, the movement requests and the location management. The rover movement is divided in steps with a specific duration. For example, if the rover needs to rotate 45 degrees, the module will not keep the rover rotating until it has achieved the new orientation but it divides the angle into steps of the maximum duration (set to 2 seconds). If the angular speed of the rover is set to 4.5 degrees per second, the rover divides a 45 degrees rotation into 5 steps of 9 degrees ($4.5deg/sec \cdot 2sec$). This is important in order to safely interrupt a movement and to update the rover position between steps.

- *Drill*: this module controls the drill electronic box and the drill tool. Its functionality is to place the drill box and to perform the necessary operations to obtain a sample from the subsurface of Mars using the drill equipment. To safely operate the drill the rover locomotion subsystem must be off.

- *Cameras*: the cameras installed on the ExoMars mast are managed by this module. It accepts requests to acquire images from the NavCams, PanCams or the HCR camera, and, if it is necessary, in a desired mode (resolution or filtered for example). When we want an image from a specific position, the mast must be previously positioned to focus the point. In the next version, some AI vision algorithms will be integrated in the Cameras module (or into specific modules) to extract information from the acquired images (see section 5).

- *Mast*: the Mast module is the responsible of the rover mast placement, that is typically related to the image acquisition.

- *Communications*: this module is designed to collect the relevant data from other modules or data sent by upper layers. It relays the information to an orbiter or to the Earth using the high gain antenna. Also it is in charge of receiving the telecommands from the ground operation station and sending those to the executor.

3.2. Executive layer

The executive is the coordinator of the other two layers. The executor is connected with both, the functional layer and the deliberator, and it is reponsible to coordinate them. In addition to the coordination functionalities, the executor integrates reactive skills and failsafe routines.

The model that guides the execution flow is a set of hierarchical plans written in PLEXIL. The top PLEXIL plan is responsible of the planning/replanning processes, and the interaction between the executor and the planner control. This is associated with the high level interface adapter which allows to read the plan obtained by the deliberator and access and modify the information contained in the problem file. When the plan is valid, the Plexil Executive reads the actions one by one, and executes them. Each action must be associated with a PLEXIL node (normally is a PLEXIL library, that is, an external PLEXIL plan) that manages the correct decomposition and execution of the action.

For each possible action, there is a G^{en}oM module that executes it. Examples of possible actions obtained from the planner are rotate, drill or recharge. The rotate action correspond to a rotation request of the Locomotion module. The drill action is responsibility of the Drill module. Recharge is managed by the Power module. In order to connect each G^{en}oM module with the executor, there is a interface adapter that communicates the module with the Plexil Executive. This interface sends and monitors the requests to the G^{en}oM module, and catches the result sent by the module. If the execution is correct, the executor continues its plan without change, but if an error is reported, the PLEXIL plan is responsible of finding a solution. For example, if a minor camera failure is detected, the PLEXIL plan can skip this image acquisition, or try to change the camera mode. For locomotion problems, such as trying to cross through more complicated terrain than expected, the PLEXIL plan can stop the motion and propagate new terrain data to the deliberator in order to try to

obtain a new route. When problems are more serious, the defined behaviour in the PLEXIL plan must be to set the rover into a safe state or to wait for human intervention.

Each PLEXIL plan that carries out the execution of an action can also contains PLEXIL libraries (if needed). This allows us to expand the functionality of the executor without modifying anything else in the architecture.

3.3. Deliberator layer

The high layer or deliberator of the architecture has the function to obtain a feasible and safely route between the initial position of the rover and a final point, performing the defined science objectives during the planned trajectory. In order to describe this layer, we differentiate three elements: (i) the planner, SGplan$_6$; (ii) the PDDL files which contain the models for the rover and terrain, and the actions that the rover can perform (both, input files to the planner); and (iii) a library that manages these elements and provides a planning framework to the executor.

The first element can be dynamically replaced in execution time. The planner election is conditioned by the algorithm resolution and the PDDL version that it supports. We need a planner that can manage metrics, fluents and constraints (that is, PDDL version 2 and some elements of version 3).

The problem and domain files contain the knowledge of the rover and the actions that it can perform. The domain specifies the actions, like go ahead, rotate, drill, recharge or acquire images. These actions are high level actions and the executor is on charge of decomposing them into a valid sequence of commands that the functional layer can execute. The actions representation include the duration and the energy consumption. For both elements, the locomotion duration and consumption is based on the rover speed and the travel distance. The other actions have a fixed cost specified in the problem file. The energy is treated as a fluent, but it is not the best solution; the energy model can be more effective if it were treated as a resource (some planners that we try do not find a solution when the energy is not enough to achieve goals and the rover needs recharge).

The rover and terrain models are defined in the problem file. The terrain model is shown in Fig. 4. The environment is discretized in squares with a pre-defined size. Each square has an altitude value and a measure of the aridity of that point. The altitude and aridity are employed in the actions movements and allows the planner to find paths which are more flat and with less altitude difference than the rover can overcome. Also in each point can be fill with an obstacle, a recharge point or the ExoMars rover.

The problem file moreover, includes the model that defines the rover position, available subsystems and associated data, and the goals that we want to achieve. The ExoMars PDDL model is shown in Fig. 5. The rover

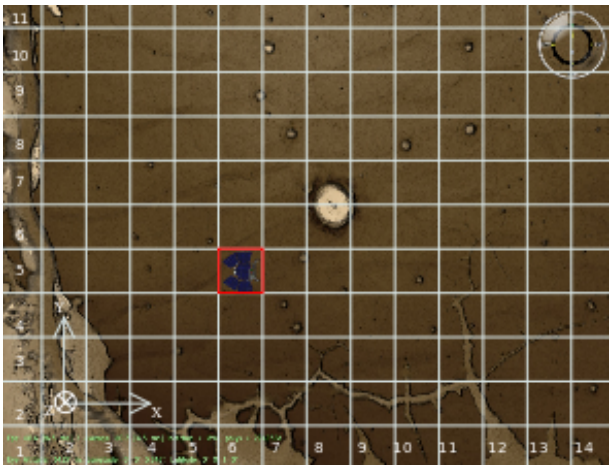

Figure 4. Zenithal vision of the high level terrain model.

model contains the initial position and orientation, and the subsystem definitions. Each subsystem is defined by a predicate to indicate that this subsystem is available, and a set of functions that define the energy consumption and operations time of the subsystems. For example, the locomotion subsystem has two speeds (one for lineal speed and other for angular speed) and another two energy consumptions in function of the type of movement. Also for this subsystem, a value that indicates the maximum altitude difference that the rover can overcome is defined. The last element of the problem is the goal(s) definition. It defines the tasks that the rover must perform, such as, go to a desired location, drill in a specific point or take an image from a location and with a particular orientation and mode. If the planner supports plan preferences, one or more targets can be not satisfied by the planner in function of a penalty defined for each goal. This implies that if a goal is too expensive to achieve and its penalty is low, the planner is allowed to skip this target in the plan.

```
(position ExoMars C5_6)
(orientation ExoMars east)
 (= (energy ExoMars) 2.15)
(has_solar_panels ExoMars)
(has_locomotion ExoMars)
 (= (lineal_speed ExoMars) 20)
 (= (energy_per_cm ExoMars) 0.000019843)
 (= (angular_speed ExoMars) 4.5)
 (= (energy_per_deg ExoMars) 0.00007716)
 (= (max_gap ExoMars) 20)
(camera_mode ExoMars NavCam lowRes)
(camera_mode ExoMars PanCam lowRes)
(camera_mode ExoMars PanCam highRes)
 (= (cam_energy ExoMars PanCam) 0.002)
 (= (cam_energy ExoMars NavCam) 0.001788)
 (= (time_to_picture ExoMars) 10)
(has_drill ExoMars)
 (= (drill_energy ExoMars) 0.007715472)
 (= (time_to_drill ExoMars) 35)
```

Figure 5. ExoMars PDDL model.

Finally, the library that controls both the execution of the planner and the acces to the problem data is designed to request for planning or replanning, to read the obtained plan (actions and their objects) and access the problem file in order to read data (object, predicates or functions values) or to modify the existing information (add, replace or delete). This library is connected with the executor through the correspondent interface adapter.

4. CONCLUSION

In this paper we have presented a revision of our previous control architecture (used in the Ptinto robot) adapted to the ExoMars rover. The design philosophy used (based on general purpose systems), has allowed us to adapt and improve the models of each layer according to the necessities of the rover. To our initial functional layer, we have add G^{en}oM modules. This implies that the hardware support can be updated with less cost than codify a functional layer in a generic programming language. Also, G^{en}oM allows us to encapsulate each subsystem of the rover in a module, whose functionalities could be incrementally fullfilled and tested, decreasing the necessary time to set up an early version of the system.

Since both, the high level layer and the executor are made by general purpose systems, this work can be focused on the study of the ExoMars subsystems, the connection with the Simsat Generic Controller, and further, in the model design of each layer. However, an important conclusion we have reached is that the models of each layer are strongly dependent on the attached layers, that is specially significant for the power model. This involves some problems due to the different vision of the world that each layer has. In our case, the G^{en}oM module in charge that the power subsystem has a power model based of the instant consumption of each subsystem power status (off, standby, heating, etc.), and that the PDDL model has a less precision model that manages full operations (drill, move two meters, etc.).

Therefore, we believe that this architecture can be a starting point for a more complex control system that will be based on the design of high level models rather than on the underlying implementation issues.

5. FUTURE WORK

There is still work to be done to provide full autonomy to the architecture. Our first step will be to implement an artificial vision system that uses the stereo vision cameras of the rover in order to obtain information of the terrain. This information must be analyzed and classified to find the interest elements of the scene and their positions. With this data we can update the information of the high level terrain model (that is, the PDDL problem) and to eliminate the need to manually enter all the terrain information.

In the long term, we are interested in including multi-agent capabilities in the architecture with the purpose of controlling a group of robots that interact and collaborate to achieve common goals.

ACKNOWLEDGMENTS

This work is funded by the Castilla-La Mancha project PEII09-0266-6640.

REFERENCES

[1] Muñoz, P., R-Moreno, M. D. & Castaño. B. *Integrating a PDDL-based planner and a PLEXIL-Executor into the Ptinto Robot*, In Proceedings of the 23rd International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems: Next-Generation Applied Intelligence (IEA-AIE 2010), Lecture Notes In Artificial Intelligence. Crdoba, Spain, June 2010.

[2] Mallet, A., Fleury, S. & Bruyninckx, H. *A specification of generic robotics software components: future evolutions of G^{en}oM in the Orocos context*. In International Conference on Intelligent Robotics and Systems, 2002, Lausanne, Switzerland.

[3] Verma, V., Jónsson, A., Pasareanu, C. & Iatauro, M. *Universal Executive and PLEXIL: Engine and Language for Robust Spacecraft Control and Operations*, In Procs. of the American Institute of Aeronautics and Astronautics Space 2006 Conference, Sept. 2006, San Jose, CA, USA.

[4] Gerevini, A. & Long, D. *Plan Constraints and Preferences in PDDL3*, Tech. Rep., Dept. of Electronics for Automation, University of Brescia, Aug. 2005, Italy.

[5] Hsu, C.W. & Wah, B.W. *The SGPlan Planning System in IPC-6*, Sixth International Planning Competition, Sept. 2008, Sydney, Australia.

[6] Poulakis, P., Joudrier, L., Wailliez S. & Kapellos, K. *3DROV: A Planetary Rover System Design, Simulation and Verification Tool*, In Procs. of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS), February 2008, Hollywood, USA.

[7] E-Martín, Y., R-Moreno, M.D. & Castaño, B. *PIPSS*: A System based on Temporal Estimates*, Thirtieth SGAI International Conference on Artificial Intelligence, December 2010, Cambridge, UK.