Advanced Design and Implementation of a Control Architecture for Long Range Autonomous Planetary Rovers

A. Martin-Alvarez [1], S. Hayati, R. Volpe, R. Petras

Autonomy and Control Section
Jet Propulsion Laboratory / California Institute of Technology
Mail Stop 198-219
4800 Oak Grove Drive,
Pasadena, CA 91109
e-mail: alex@telerobotics.jpl.nasa.gov
Tel: (818) 354-4725 Fax: (818) 393-5007

ABSTRACT

An advanced design and implementation of a Control Architecture for Long Range Autonomous Planetary Rovers is presented using a hierarchical top-down task decomposition, and the common structure of each design is presented based on feedback control theory. Graphical programming is presented as a common intuitive languague for the design when a large design team is composed of managers, architecture designers, engineers, programmers, and maintenance personnel. The whole design of the control architecture consists in the classic control concepts of cyclic data processing and event-driven reaction to achieve all the reasoning and behaviors needed. For this purpose, a commercial graphical tool is presented that includes the mentioned control capabilities. Messages queues are used for inter-communication among control functions, allowing Artificial Intelligence (AI) reasoning techniques based on queue manipulation. Experimental results show a highly autonomous control system running in real time on top the JPL micro-rover Rocky 7 controlling simultanously several robotic devices. This paper validates the sinergy between Artificial Intelligence and classic control concepts in having an advanced Control Architecture for Long Range Autonomous Planetary Rovers

1. Introduction

A <u>highly autonomous rover</u> is desired in a <u>planetary exploration mission</u>. The human on ground only needs appropriately abstracted state and status feedback in the telemetry down link. Thus, time delays and temporary loss of communications to Earth are not a problem because no real time control loops are closed via the up/down link. This also reduces the power consumption of rover subsystems like telecommunication and makes more resources available for the actual locomotion.

Control Architectures are key elements to identify, define, and implement all the control aspects needed to achieve the requiered autonomy. Then from the last decade, definition and design of <u>control architectures</u> have been of major interest in the research field of autonomous mobile robots [7][10][11][12][19] but however very little have been done for a high challenging scenario. This paper is pionner in suggesting <u>tools and procedures to design, implement, and maintain</u> control architectures for a highly complex mobile robot as a future <u>Long Range Autonomous Planetary Rover</u> [1][23].

[1] Visiting scientist at JPL

This paper covers three phases of building a complete control architecture for Long Range Autonomous Space Rovers. The first phase is to build a Functional Control Architecture defining what are the needed control functions and interactions among them. Second phase is to build an Operational Control Architecture defining where all these control functions (either on-board or on-ground) are located to achieve the desired control autonomy for a space scenario. An Operational Control Architecture also defines extra operational support functions in addition to the control ones. The last design phase is to build an Implementation Control Architecture defining how to implement (software, hardware, or human intervention) all the control and operation support functions.

This paper presents frameworks, procedures, and an advanced tool to support the design of each type of control architecture and the transition among them for a Long Range Autonomous Space Rover. We used a graphical programming for the design of the control architecture as a common and intuitive languague for a design team composed of managers, architecture designers, engineers, programmers, and maintenance personnel. Graphical editors of Control Shell [8] are used for such graphical programming. In addition, Control Shell provides a real-time software framework for the implementation, debugging, maintenance of all control functions and interactions among them defined in the design phase.

Chapter 2 describes the Functional Control Architecture for a Long Range Scientific Mars Rover, defining first an Integrated Control Architecture (ICA) to put the Planetary Rover Control Architecture into the context of the overal planetary exploration scenario. Later objectives and State of the Art of Control Architectures for Mobile Robots are given and the MObile Robot Control Architecture (MORCA) is defined as the baseline for the Functional Control Architecture of a Long Range Scientific Mars Rover. Chapter 3 describes the Operational Control Architecture for a highly and realistic autonomous planetary rover. Finite State Machines and Data Flow Diagrams are shown for the design of this architecture using Control Shell toolkit. Chapter 4 describes the basic approach for the Implementation Control Architecture and Chapter 5 gives experimental result with the JPL micro-rover rocky7 when executing autonomously a high level task.

2. Definition of the Functional Control Architecture for a Space Rover

The Functional Control Architecture specifies what are the needed control functions and interactions among them. Creating this architecture is the first phase of the design.

2.1 Integrated Control Architecture (ICA)

Following the top-down design approach, we first define the Integrated Control Architecture (ICA) to support general space scenarios dealing with several cooperative elements such as robotic arms, orbiters, landers, and planetary rovers. The major characteristics of ICA are:

- contains the functional control architectures of all the space elements that take part fully or partially of a space mission,
- a common mission layer is defined on top of all the element control architectures. The common mission layer has as input a common mission that is decomposed (planned or scheduled) into task commands for each space element. The global mission layer controls the execution of the space mission, for example controlling the sequence of tasks (dispatching),
- allows the placement of control functions of one space element on different space element in order to achieve optimal overall performance,
- shows the communication among the control functions.

The definition of these interactions among control functions in different elements play an important role in a space mission based on planetary rovers. Interactions are needed when some elements are in a better position to carry out control functions of other elements. For example a planetary rover is in better position than a lander to carry out landing functions in real time, uplinking either relative positions and attitude between lander and landing site or landing trajectories.

The main apportation of the Integrated Control Architecture (ICA) to this paper is to emphasize that the functional control architecture of a planetary rover has to be open to interact with other space elements.

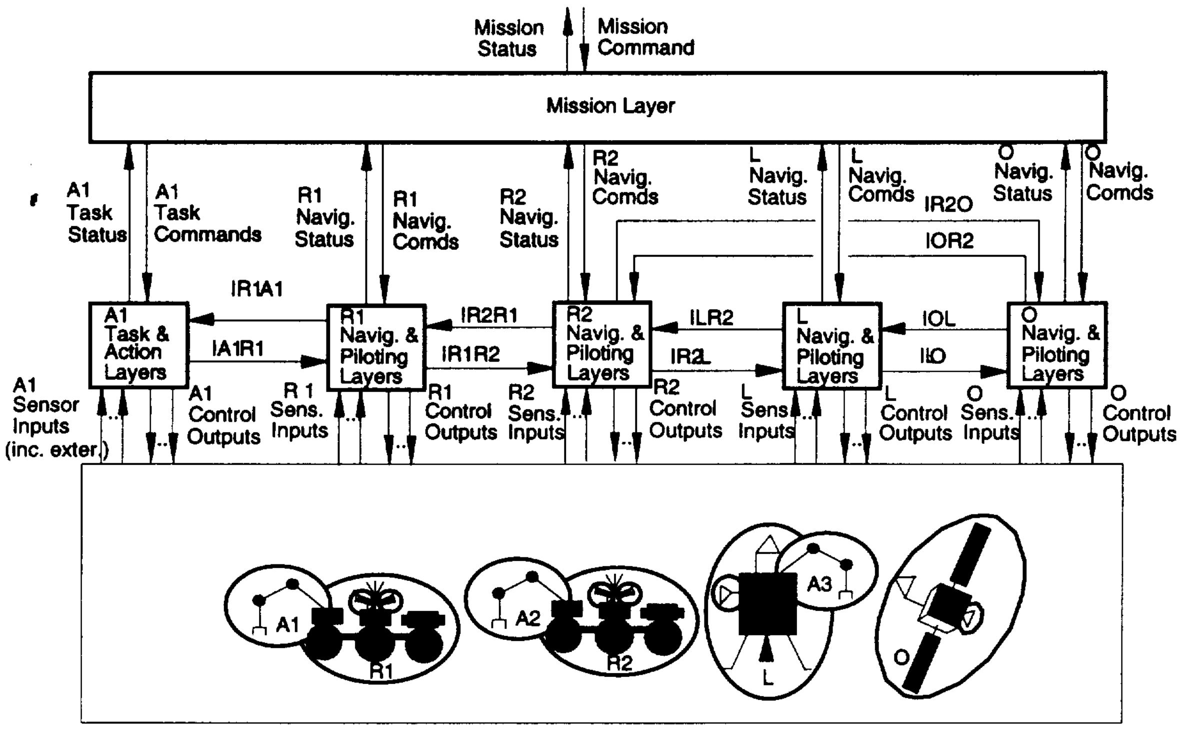

Fig. 2. Integrated Control Architecture (ICA)

2.2 Objectives and State of the Art

The main objectives of a Functional Control Architecture for a planetary rover are:

- Generality: To be a general framework to allow a better understanding of all the functions needed to achieve the design requirements of a control system, in order to fulfill a specific mission (see Table 1), independent of operations and implementation features (control hardware, control software, and human intervention),
- Inter-Element Cooperation: To support the interaction among space elements (several planetary rovers, robotic arms, orbiters, and landers) taking part of a mission, identifying what kind of information or commands are exchanged among such elements.
- Unity: To unify in the same framework all the possible implementations of Rover Control Systems.
- Flexibility: To allow assessment of the capabilities and performances for each control implementation as well as for different configurations of cooperation among the control architectures of the elements taking part of a mission.
- Robustness: To allow robustness both in the presence of uncertainties about the knowledge of the environment, and in the presence of inaccuracies and limited performances in rover, sensors, and actuators,

- Quickness: To allow execution in real time,
- Savings: To achieve low cost

Mission Commands	Definition	Parameters
INSPECT_AREA	Build a topographic map of the environment.	<initial position> <surface to inspect> <final location>
SURVEY_SPACECRAFT/ ROVER	Survey lander, parachutes (Mars), aeroshell shield (Mars) or other rover	<spacecraft/rover location> <area to survey>
RELAY_DATA	Relay data from on-ground space element (e.g. rover or lander) to ground segment (e.g. Earth)	<location of ground space element> <location of ground segment >
INSTALL_INSTRUMENT	Deploy and install instruments.	<deployment location> <type of instrument>
COLLECT_SAMPLE	Collect, retrieve or analyse soil or rock samples.	<sample location> <type of sample> <amount of sample> <rendez-vous point to return samples>
RETRIEVE_ITEM	Off loading of cargo items from lander	<lander location> <type of item> <placement location>

Table 1. Space Rover Mission Commands

There are a great number of approaches for the design of planetary rover control architectures without a common definition of control functions and terminology [10][11][12][19]. Even when each mobile robot application and scenario could require specific control functions and interaction mechanisms among them, a common functional control architecture is of great interest for the sake of a better understanding and assessment of capabilities. Also, a complete Functional Control Architecture for a future Long Range Autonomous Space Rover is mandatory to handle the control complexity that was never required in previous spacecrafts [24][26].

Three main approaches for functional control architectures have been defined in the last ten years: hierarchical, behavioral, and a hybrid of both.

Pure Hierachical Architectures. In this organization scheme based on a hierarchy of layers, decision making processes are present at each resolution/abstraction level to either generate action commands to the lower adjacent level or perceptual information to the upper one [10] [18] [19].

Pure Behavioral Architectures. The central idea of behavioral architectures is that a control system consists of the desired external manifestations or behaviors of the system. There is not any goal decomposition at execution time. This is done in the architecture design since goals do not change from one problem to the next [11] [12].

Hybrid Architectures. These architectures try to combine hierarchical and behavioral approaches. Hybrid architectures are a consequence of either an evolution of behavioral architectures or the definition of a new architecture taking the most relevant advantages of both approaches [2] [16].

Pure hierarchical architectures are good to define clearly all the control functions needed but fail in real time execution having too much high level reasoning even without dealing with all the uncertainties and constraints of a real rover application. Pure behavioral architectures are good in real time execution reacting rapidly with the environment without high level reasoning but fail both when the intelligence required is high and in a very complex scenario. Hybrid architectures are the answer to our scenario and several designs have been done but however all of them were used in simple scenarios. Our goal is to define a hybrid control architecture that combine properly all the advantages of control and AI techniques for a Long Range Scientific Mars Rover working in a very complex planetary rover exploration scenario.

2.3 MObile Robot Control Architecture (MORCA)

To accomplish all these objectives, the MObile Robot Control Architecture (MORCA) [2] was defined as a general framework, mapping well-known approaches from the literature [10] [11] [12] [16] [18] [19]. MORCA proved that behavioral and hierarchical approaches are not in conflict. The only difference is whether control functions are used in execution or in preparation.

For the design of a new functional control architecture for a Long Range Autonomous Space Rover, we use the top-down task decomposition engineering approach followed in the hierarchical Mobile Robot Control Architecture MORCA [2] [5]. This decomposition of the problem into lower level functions together with an exhaustive definition of the commands and data among them, allows a better understanding, test and update of a complex rover control architecture. The levels of the MORCA architecture correspond to a subsequent refinement of commands to the mobile robot, from highest level mission commands via commands on navigation, piloting, wheel motion coordination, to individual wheel control commands.

The sequence of different tasks or mobile robot command levels has been identified, increasing in complexity and abstraction, as it is shown in the following table.

M.R.command levels	Mobile Robot commands	Similarity with human commands
Degree 6 (Mission Comds)	a) Inspect area b) Collect sample	High level of messages between two people (the boss to his employee)
Degree 5 (Navigation Commands)	a) Go to a location	Commands from a person, in an unknown city to a taxi driver to go to a specific address
Degree 4 (Piloting Comds)	a) Stay in a direction until event b) Follow/Reach an object until event	Commands to a car driver to reach an address expressed by an occupant who knows how to go
Degree 3 (Trajectory Commands)	a) Border/Follow object b) Go straight object c) Stay in direction e) Reduce/Increase speed	Driver commands to a learner
Degree 2 (DeviceControl Cmds)	a) Steering Angle b) Speed	Car driver control actions
Degree 1 (Control Outputs)	a) 5 Volts to Motor 1 b) Switch on a brake	Electrical Signal from the nervous system to muscles

Table 2. Mobile Robot Command levels.

Table 2 shows a hierarchy in mobile robot commands where all commands of degree *i* can be transformed into a set of commands of degree *i-1*. The decomposition and the control of such commands are performed by a set of functions contained in a control architecture. Because of the mentioned hierarchy in mobile robot commands, our approach for the definition of MObile Robot Control Architecture (MORCA) also follows a hierarchical structure, based on different layers. The more the layers are able to work autonomously (no human presence), the more complicated are the tasks which the rover achieve itself.
MORCA is structured into a hierarchy of functional layers where each layer is structured into three parallel functional branches based on the concept of feedback control [7] [17]:

• Forward Control (FC). Responsible for activity decomposition, execution planning, control, and command dispatching.
• Nominal Feedback (NF). Functions for refinement and update of a priori knowledge ("world models") based on the actual, but essentially expected, evolution of the process and consequently formulation of controlled adjustments of the FC.
• Non-Nominal Feedback (NNF) takes care of the correct functioning of this layer, detecting and analysing non-nominal situations. It contains functions for the monitoring of discrepancies between actual and allowable states in both the FC and the NF functions, diagnosis of their origins, and generation of directives and constraints for FC.

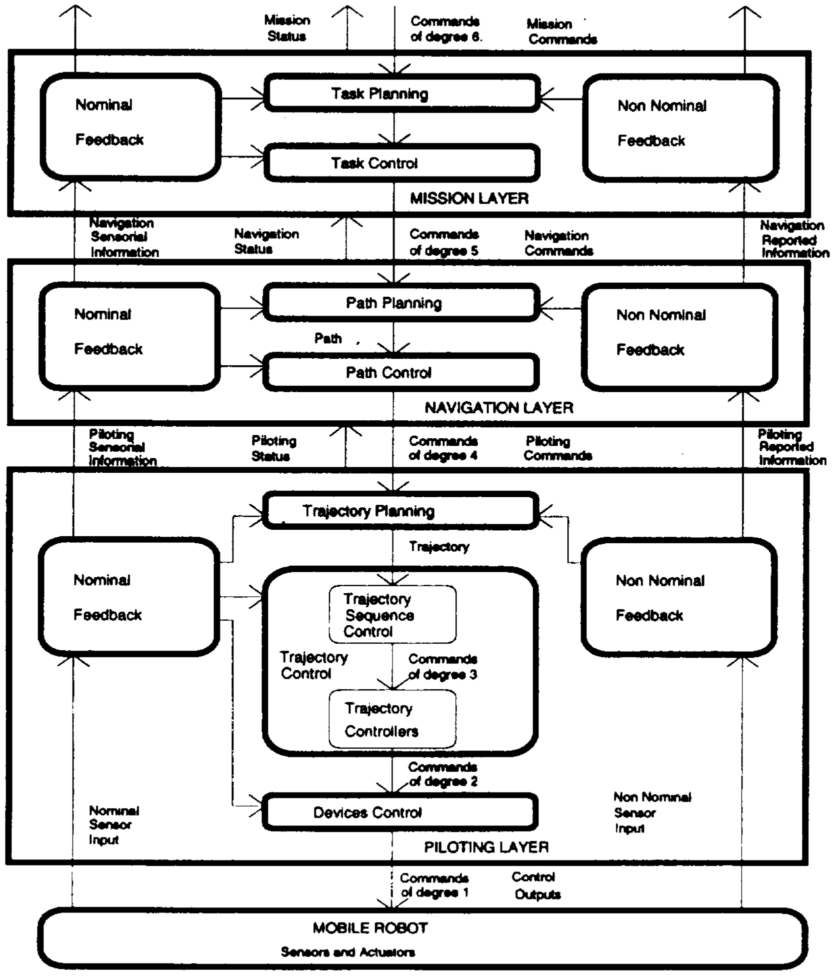

Figure 1. Mobile Robot Control Architecture.

An example of the Forward Control activity decomposition and planning is given in the appendix A where: navigation nominal feedback updates the map and localizes the mobile robot, destination, places of interest, and path segments inside the map; navigation non nominal feedback detects when the mobile robot gets lost, generates strategies to find references for localization, and deals with unexpected situations reported by the pilot; piloting nominal feedback detects environment features, estimates trajectory status and rover internal state, and provides world model updates to navigation; and piloting non-nominal feedback detects hazards, for example unexpected rocks or loose sand, and generates the needed recovery strategies. Due to the modulatity of the architecture in layers and functions, fast execution is possible using parallel computation.

A similar structure of MORCA can be used for the control system of other elements, such as robotic arms or spacecrafts, taking part of the same mission. Also inside each element interaction among different S/Ss (e.g. motion, thermal, and power control) is supported. In this way, MORCA also supports the interaction among control functions in different mission elements. each control architecture in ICA is based on MORCA as will be explained next,

2.4. Functional Control Architecture for a Long Range Scientific Mars Rover

A Long Range Mars Rover is a complex spacecraft containing a set of elements, such as a mobile platform, manipulator, pointable cameras, and scientific instruments, where each of these elements has its own control system (see Figure 3).

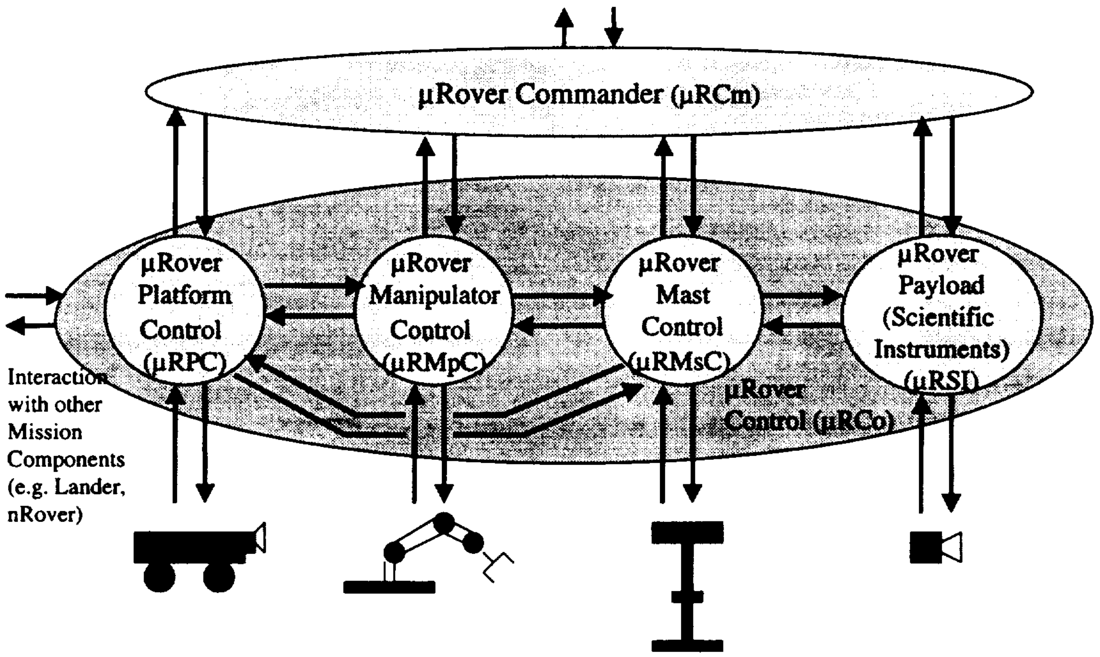
µRover Commander (µRCm)
µRover Platform Control (µRPC)
µRover Manipulator Control (µRMpC)
µRover Mast Control (µRMsC)
µRover Payload (Scientific Instruments) (µRSI)
µRover Control (µRCo)
Interaction with other Mission Components (e.g. Lander, nRover)

Figure 3. Space Elements of a Long Range Scientific Mars Rover

In addition, as a regular spacecraft, each control system is divided into several subsystems, such as, locomotion/propulsion, thermal, power, and telecommuniation (see Fig. 4).

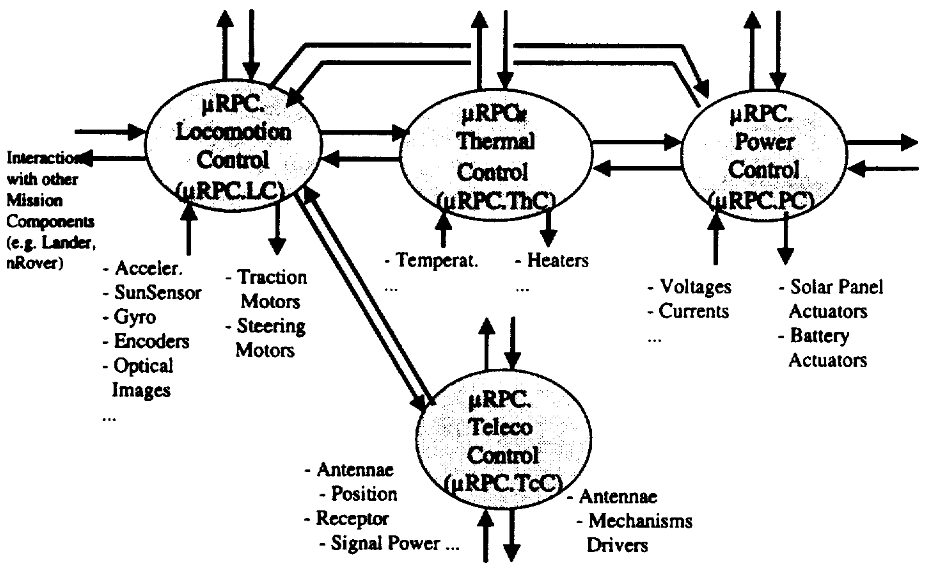

Figure 4. Spacecraft Control Subsytem

Following an ICA structure, a centralized mission control layer, called micro-Rover Commander for a Mars rover scenario, has to be added to command and to coordinate all the rover elements.

Following the MORCA and ICA design philosophy, first an exhaustive definition of the commands and data transfer among rover elements [1] [20] is required to define the Functional Control Architecture for a Long Range Scientific Mars Rover. Table 3 shows the set of Sojourner and Rocky 7 Body Motion Commands as an example.

Body Motion Control	
Sojourner	Rocky 7
Capture_Image_with_Camera <c> at_Exposure <t>, Return_Region_from (r1,c1) to (r2,c2) with APID <id>	Pilot_update <x,y,z,theta> (in panorama frame, degrees from North)
Go_to_Waypoint_at <x,y> within <m> Minutes	Pilot_goto <x,y> (in panorama frame)
Material_Adherence	Pilot_goto_direct <x,y> (in panorama frame)
Move_Backward <n> Counts	Pilot_head <theta,x> (degrees from North, distance)
Move_Forward <n> Counts	Pilot_head_direct <theta,x>
Set_Parameter <Maximum squeeze mode navigation> = <Value>	Pilot_face <x,y> (in panorama frame)
Set_Parameter <Traverse cycle distance> = <Value>	
Turn_Left <n> Bams	
Turn_Right <n> Bams	
Turn_to_Heading <h>	
Turn_Towards <x,y>	

Table 3. Sojourner and Rocky 7 Body Motion Commands

Once all the Commands were defined for each element control system and for its subsystem, then all the control functions (see Table 4) are identified following the hierarchical principle of MORCA.

Micro-Rover Control	Micro-Rover Commander	Platform Control	Locomotion Control	Navigation Layer		Path Panning + Control
				Piloting Layer	Body Layer	Planning + Control
					Wheel Coordination Level	Planning + Control
					Wheel Level	Planning + Control
					Actuator & Sensor Level	Planning + Control
			Thermal Control	Actuator & Sensor Level		
			Power Control	Actuator & Sensor Level		
			Telecom Control	Actuator & Sensor Level		
		Manipulator Control	Task Layer	Planning + Control		
			Action Layer	Planning + Control		
			Actuator & Sensor Level			
		Mast Control	Task Layer	Planning + Control		
			Action Layer	Planning + Control		
			Actuator & Sensor Level			
		Payload Control	Actuator & Sensor Level			

Table 4. Control Functions for a Long Range Scientific Mars Rover

However, there are rover element control S/Ss (e.g. mobile platform locomotion S/S) that are more complex than others needing a higher refinement of commands. Here again, MORCA design philosophy are succesfuly applied.

3. Design of the Operational Control Architecture for a Space Long Range Science Rover

Once the needed control functions and interactions among them are defined in the Functional Control Architecture an Operational Control Architecture[2] defines where all these control functions are located (either on-board or on-ground) to achieve the desired control autonomy for a space scenario. An Operational Control Architecture also defines extra operational support functions (see Fig. 5).

[2] Notation: The two parallel lines for data store, discontinous line for syncronization signal and arrow for data flow.

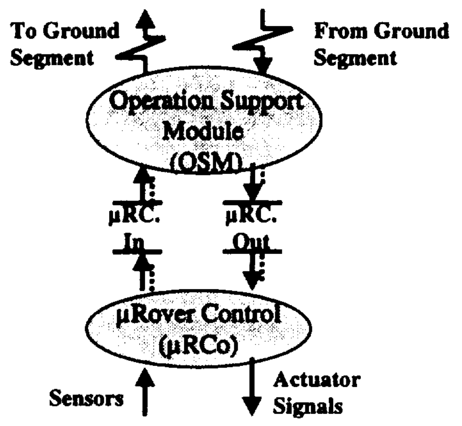

Figure 5. Operational Control

3.1. Commands and Information Transfer

Commands and information transfer are basically designed as single messages and queues where last ones require extra pointers, e.g. a waiting and a execution pointer. Two main types of queues have been implemented, FIFOs (First Input First Output) and LIFOs (Last Input First Output). The first type was the one mostly used and the last one was mainly used for non-nominal event communication. In addition, these queues support Artificial Intelligence reasoning techniques based on queue manipulation.

Figure 6 shows this inter-communication in a control/planning module. The two parallel lines include the name of the information to transfer (e.g. Data and Commands Parameters) and its associated discontinous line is a signal or stimulus to represent an event (e.g. a Command). This information is stored in FIFO queues except the non-nominal ones (Input 1.4 & Output 1.4) that follow a LIFO structure.

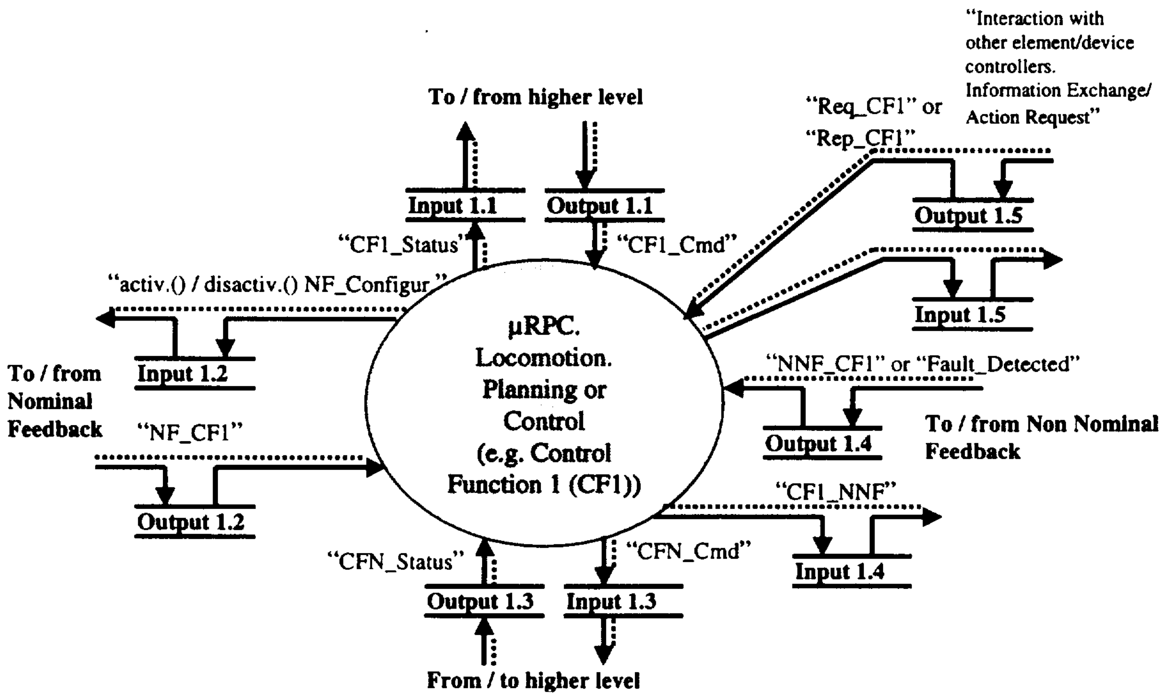
"Interaction with other element/device controllers. Information Exchange/ Action Request"
To / from higher level
"Req_CF1" or "Rep_CF1"
Input 1.1
Output 1.1
Output 1.5
"CF1_Status"
"CF1_Cmd"
"activ.() / disactiv.() NF_Configur."
Input 1.5
μRPC. Locomotion. Planning or Control (e.g. Control Function 1 (CF1))
To / from Nominal Feedback
Input 1.2
"NNF_CF1" or "Fault_Detected"
"NF_CF1"
Output 1.4
To / from Non Nominal Feedback
Output 1.2
"CF1_NNF"
"CFN_Status"
"CFN_Cmd"
Input 1.4
Output 1.3
Input 1.3
From / to higher level

Figure 6. Intercommunications in a Control/Planning Function

The complete set of on-board Control and Operation Support functions for a Space Long Range Science Rover [1][20] are given in the following table using the MORCA design philosophy.

Operational Control Architecture	Operation Support Module	Command and Telemetry Management	Command Mangement
			Telemetry Management
		Control Operation Supervisor	
		Logistic Module	
		System Administrator	
	Micro-Rover Control (see Table 4)		

Table 5. On-board Rover functions for the Operational Control Architecture

3.2. Capabilities needed for the Operational Control Architecture of a Space Long Range Science Rover

A highly autonomous rover is mandatory for Space Long Range Science Missions. With more on-board control functions there is a greater possibility that the mission goals will be reached. However, on-board control functions (including sensors and actuators) can fail requiring external help. Therefore flexible placement of operation support and control functions are needed between the Ground and Flight Segments to reconfigure the Operational Control Architecture.

In addition, this architecture reconfiguration is needed for different mission phases. For example, safety is the main issue at the beginning of a space mission, placing most of control functions on the ground segment with highly human intervention (human on the control loop). However, at the end of the mission, more challenging and highly autonomous rover tasks will be commanded having most of the control functions executing on-board.

Architecture configuration is also needed to handle anomalies like software malfunctions or loss of communication, where on-ground control functions are completely useless. Therefore, the rover automatically has to reconfigure its Operational Control Architecture in order to become more autonomous and find recovery strategies to establish communication to Earth.

As a conclusion, a Space Long Range Science Rover must have the capability to receive high and low level commands.

For a spacecraft, and then for a space rover, there are four major Operation Support Functions: Command and Telemetry Management (CTM); Control Operation Supervisor (COS); Logistic Module (LM); and System Administrator (SA), as shown in Figure 7.

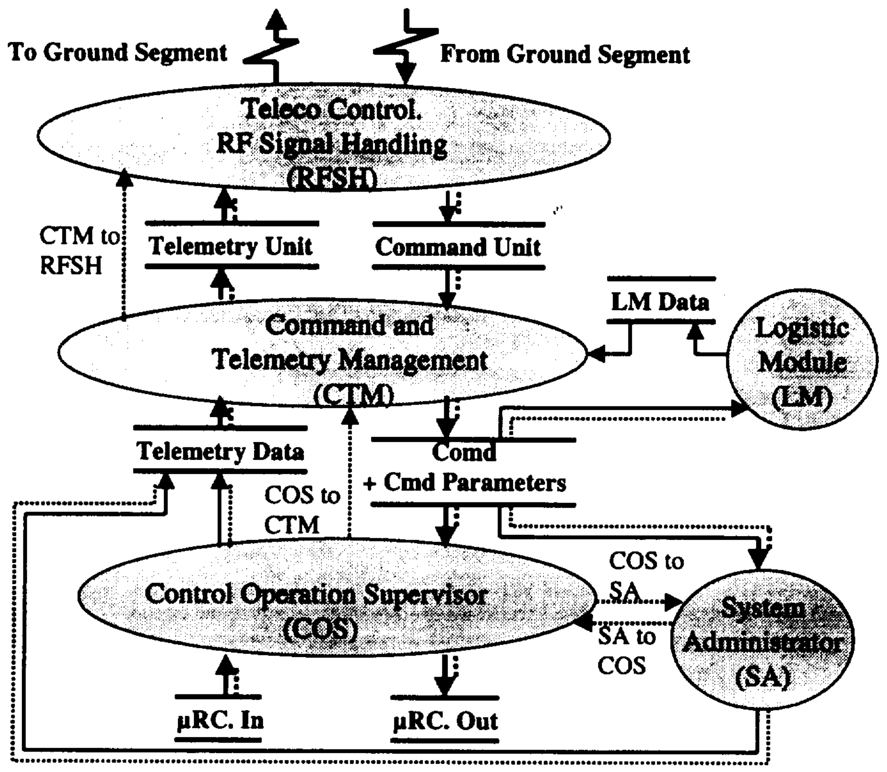

Figure 7. Main Operation Support Functions for a Spacecraft

The Command Management Module distributes the commands received from ground to the Telemetry Management Module, LM, and SA. The Telemetry Management Module handles the downlink of all the telemetry data accumulated in its input buffer. It also generates a heartbeat, and establishes telecommunication signals to Earth. See Figure 8.

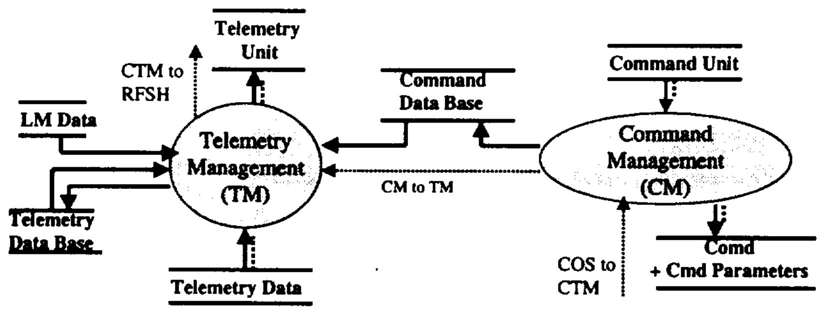

Figure 8. Structure of the Command and Telemetry Management

The Logistic Module (LM) contains the rover clock and computer hardware supervisor. It sends clock signals to other modules, for example a wake up signal. System Administrator (SA) is in charge of the file system making sure that there is enough computing resources available for the execution of operation support and control functions.

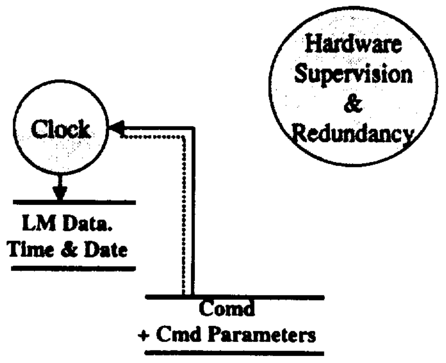

Fig. 9. Main Functions of the Logistic Module (LM)

The Control Operation Supervisor (COS) is in charge of: resources management and health care; distribution of the received commands from CM to any control function checking they are consistent with the current operation mode; reconfiguration of Operational Control Architecture when both anomalies occur like loss of communication to Earth and wake up signal is received when the rover was in "stand by" (e.g. during the night).

3.3 Design using Graphical Programming, Classic Control Concepts, and Artificial Intelligence Concepts.

The design of an Operational Control Architecture for a Planetary Long Range Science Rover [1][23] needs of a real-time framework for engineering control design dealing with the complexity of: large projects, real-time software, event processing, feedback control, and interacting teams of programmers, engineers, managers, and maintenance personnel. Therefore we use graphical programming for our design promoting understandability and as a consequence the design is quicker to learn, faster to develop, easier to debug, and less costly to maintain.

We use the well-known classic control concepts of Finite State Machine, Dataflow Diagrams, and classic queue manipulation for the design of the whole architecture. Finite State Machines are used for event-driven reaction and Dataflow Diagrams for synchronous cyclic data processing including the implementation of AI behaviors.

3.3.1. Guidelines for the use of Finite State Machines and Dataflow Diagrams

An Operational Control Architecture must be designed to work in real time, that is, fast command execution and fast reaction to events stablishing priorities among them. It is also mandatory for inter-communication efficiency and good controlability of the functioning of the Operational Control Architecture, to minimize the number of control functions working in parallel, that is, to minimize the number of Finite State Machines and Dataflow processes (both Operating System processes) but still keeping the required real time performance. Thus, it was found convenient to design a Standard Finite State Machine structure that group several control functions that were already defined in the Functional Control Architecture.

We group Planning, Control, or Dispatching functions together with Nominal, Non-Nominal, and External Event Handling functions for any architecture layer (see Figure 10). A subset of this structure is also used for Operation support functions.

This grouping was selected because these functions are exclusive, that is only one is requiered to work at once and because this grouping contains the mechanisms that establishes the desired priority in their execution. For example, whenever a diagnosis and recovery strategy generation function takes place only its execution is requiered at its layer, without considering the execution of nominal planning or any nominal control action. At the same time, a non-nominal event has the highest priority in its layer canceling the execution of any other control function in progress.

A secondary advantage of this grouping is that all these control functions share the same inter-communication function to communicate with other Finite State Machines either in the same or different control layers via the already mentioned Queues.

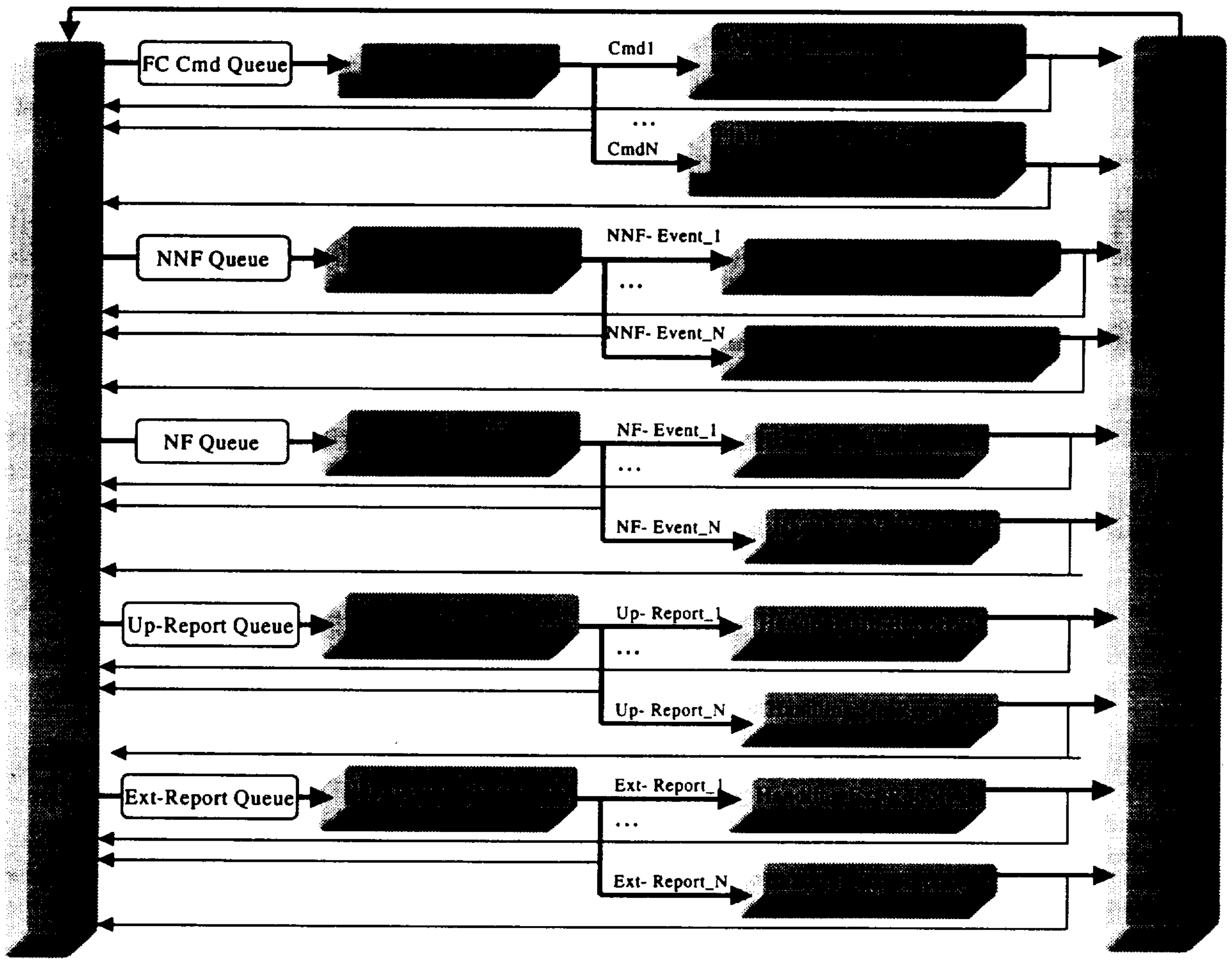

Figure 10. Standard Finite State Machine structure for Planning or Control Functions

Dataflow diagrams are used in the Operational Control Architecture for continuos cycle sequences for example as sequence composed of nominal feedback (including sampled data feedback), planning, control, and motor driver activation (see Fig. 11). Sometimes a subset of this whole sequence is only required, for example standalone Nominal Feedback functions that can run independently to detect nominal events (e.g. a trajectory final condition reached) that communicates to a Finite State Machine via its NF queue.

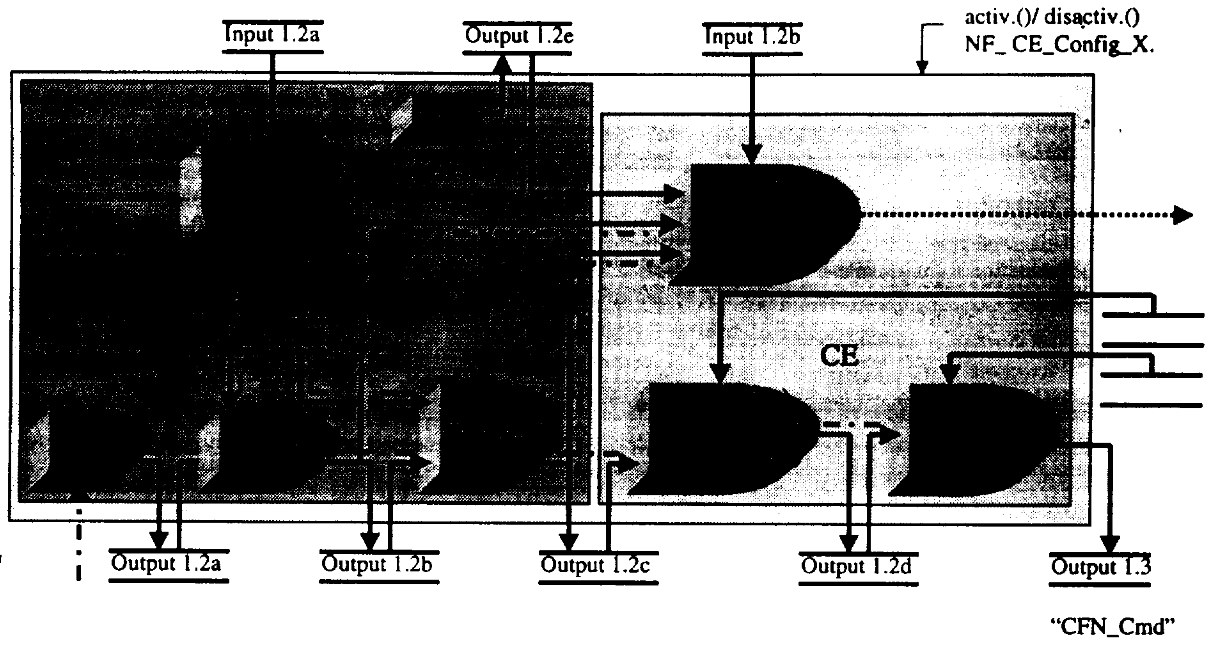

Figure 11. Schematic DFE structure for a Rover Control Continuous Cycle Sequence

Dataflow Diagrams are also used for Non Nominal Feedback monitoring, maybe sharing some components with the NF. This NNF monitoring has the capability to communicate a non-nominal event to a Finite State Machine, that is performing control or planning tasks, via its NNF Queue (see Figure 12).

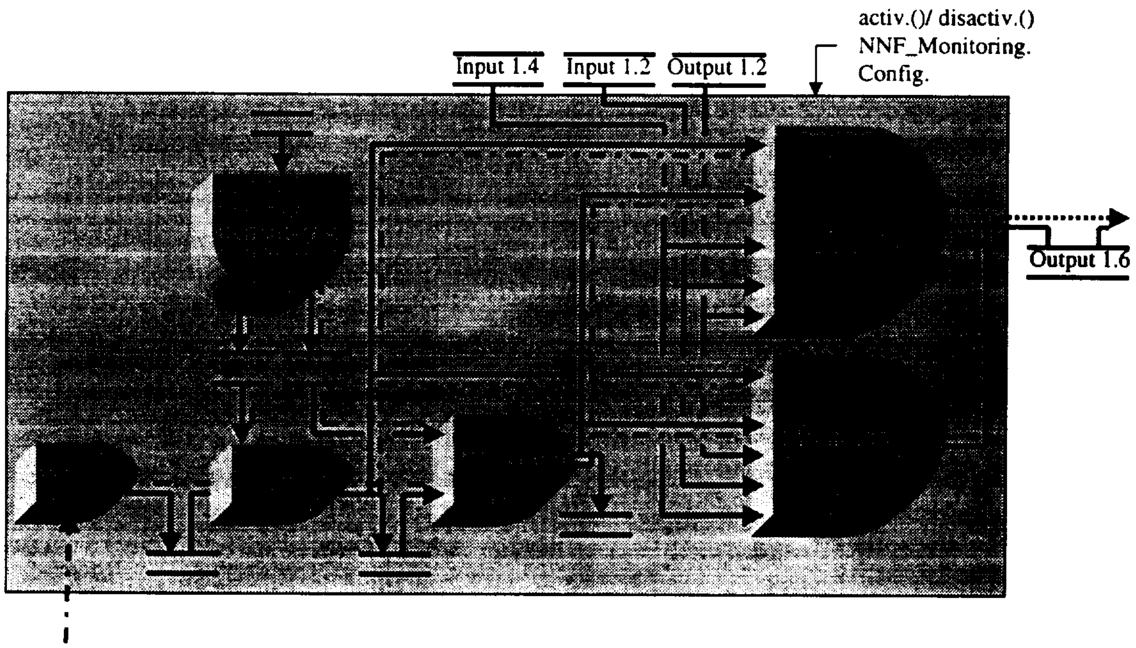

Figure 12. Schematic DFE Structure for a Non Nominal Feedback Monitoring Function.

3.3.2. Control Shell Tool

Our control architecture design uses the commercial Control Shell Tool [8] because is compliaced with our requirements of graphical programming and the concepts of Finite State Machines for event-driven reaction and Dataflow Diagrams for synchronous cyclic data processing. Control Shell also supports object-oriented modelling for the control design and implementation.

In addition, Control Shell provides system configuration control for changing operating modes and real-time matrix mathematics package (CSMat) useful for real time AI reasoning based on queue manipulation mandatory for a highly Autonomous Long Range Science Rover.

Figure 13 shows what a FSM looks like in the CS FSM graphical editor. Boxes represent states and arrows the transition among states. A name in quotes represent a stimulus, that is the event, and after the slash appears the transition function as the action after the event. The return codes of a transition function, used for decision making, select the next stay to stay waiting for a new event.

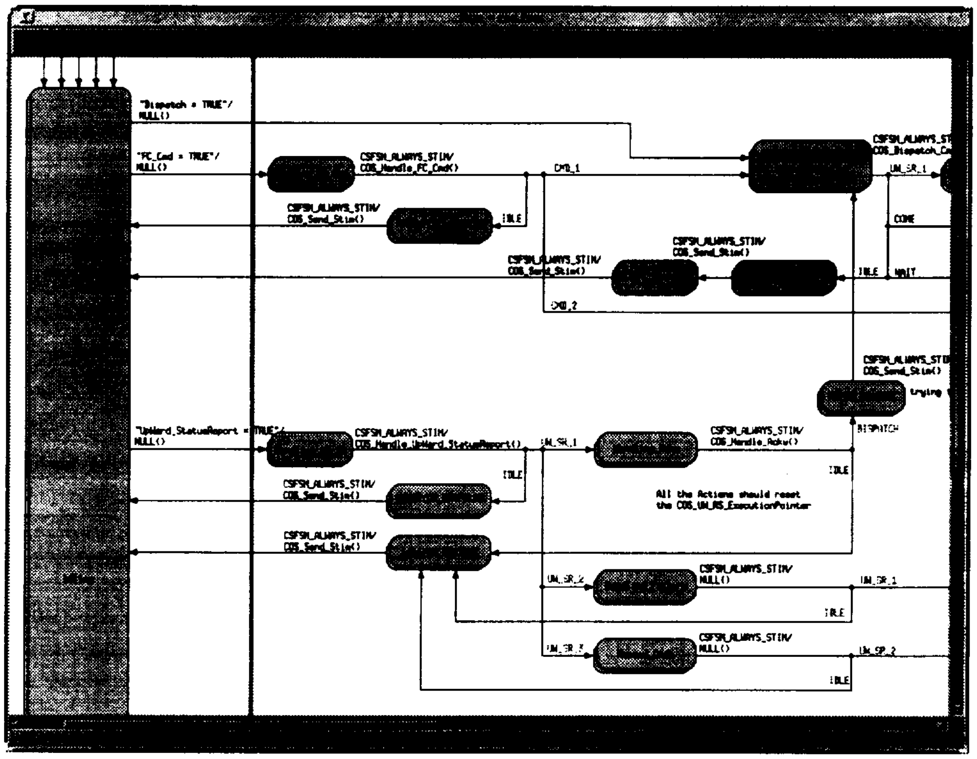

Figure 13. Example of a FSM in the Control Shell FSM Graphical Editor

Figure 14 shows how a DFE looks like in the CS DFE graphical editor. Boxes represent components, arrows represent the data flow connecting inputs and outputs (left and right side respectively) of the components, each arrow has associated the name of CSMat used a communication data, and the lines above of each component represent component's parameters mostly in CSMat format (constants are allowed as well).

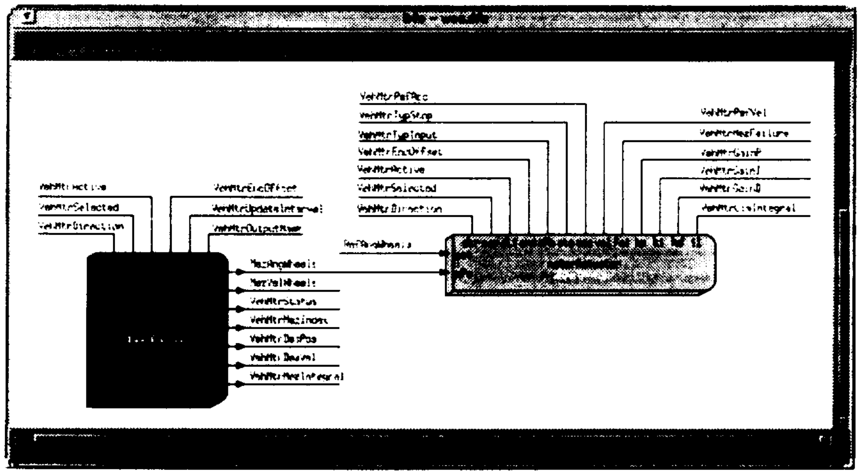

Figure 14. Example of DFE in the Control Shell DFE Graphical Editor

4. Design of the Implementation Control Architecture for a Space Long Range Science Rover

Once the Operational Control Architecture defines the location (on-board and on-ground) of control and operation control functions, Implementation Control Architecture, defines how to implement them (software, hardware, or human intervention). The well known software architecture in the computing science community, is only the software side of a Implementation Control Architecture.

For the design of this Control Architecture, we used the mentioned Control Shell tool, that using Object Oriented Programming templates, generates the C++ based structure (declaration of functions and input and output parameters) for each piece of code both for DFE components and FSM transition functions.

For our applications, we use the VxWorks Operating System running on top of a VME chassis.

5. Experimental Results

5.1 Rocky 7.

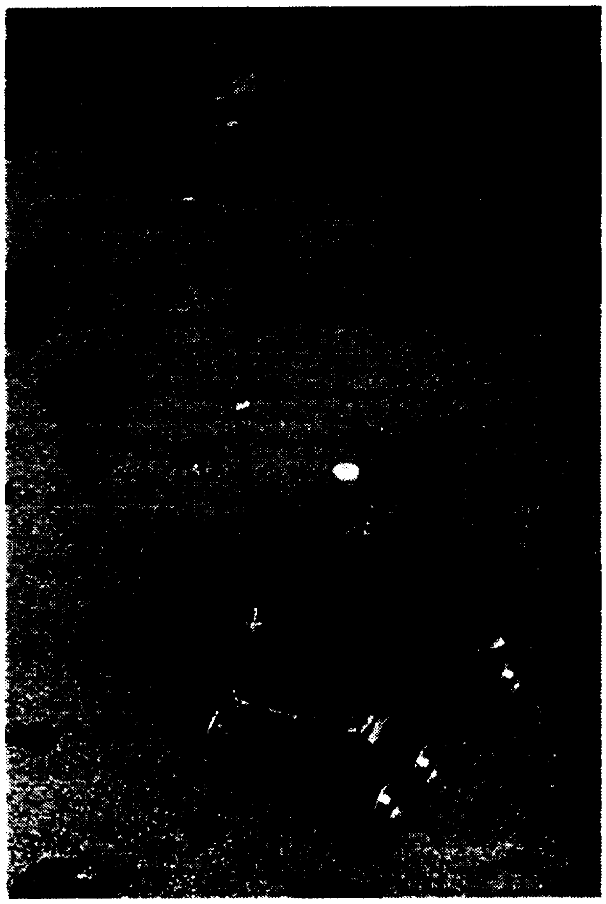
Figure 15. JPL-NASA Micro-Rover Rocky 7

The JPL-NASA micro-rover Rocky 7 [21] was used as breadboard for Control Architecture of a Long Range Autonomous Space Rover. Rocky 7 (see Fig. 15) is a research micro-rover used to demonstrate new technology concepts for use in a long range (>1 Km) traversal across Mars, scheduled for early in the next decade. Its locomotion is a modified six wheel rocker/bogey similar to Sojourner (NASA Path Finder mission).

Its main features are: 1) size: 60 x 40 x 33 cm; 2) mass: 15.7 kg; 3) power: rechargeable NiCad batteries and Si solar panel; 4) computer: 3U VME, 68060 CPU, 100 MIPS; 5) Science Payload: a selection of IR reflectance spectrometer, color filter stereo imager, multispectral close-up imager, and Mossbauer spectrometer; 5) 4 DoF Arm; and 6) a camera Mast able to deploy itself 1.4m above the ground.

5.2 Experimental results with Rocky 7 Technological Micro-Rover

Following MORCA and ICA principles, a full control system for a Long Range Autonomous Mars Rover has been designed using the graphical editors of Control Shell. As a first implementation step, a distributed task execution and a fully autonomous piloting layer have been implemented. The distributed task execution consists that both the stereo camera mast, mobile platform, and robotic arm independently execute and handle its own command queues and the rover commander coordinates all of them. The fully autonomous piloting layer achieves continuous driving without stops for obstacle detection or the planning of a new path segment. Also some operation support functions, as command management, have been implemented simulating a "real" space mission.

In the testing scenario, first the camera mast is deploied taking pictures and later a ground segment operator uses his knowledge about the rover environment and position, and the desired destination, to generate a path consisting of a set of path segments (see Appendix A for a complete example), being the operator supported with a path planner tool. The last task is when that the robotic arm takes a sample in the desired destination. In our testing scenario, navigation path planning is done off-line, by creating a set of consecutive path segments with the parameters: motion direction (forward or backward), cruise speed, final condition, final condition accuracy, and dispatching mode (syncronous or asyncronous), and stop mode after its execution.

The execution of four path segments are shown in the Figure 16, where the first two path segments (from (0,0) to (3,2) meters and from (3,2) to (5,2) meters respectively) consist in: high and slow speed forward motion respectively, the third one (from (5,2) to (4,3) meters) in low speed backward motion, and the last path segment (from (4,3) to (4,2.8) meters) in order to face the sample to observe consists in a very slow forward motion.

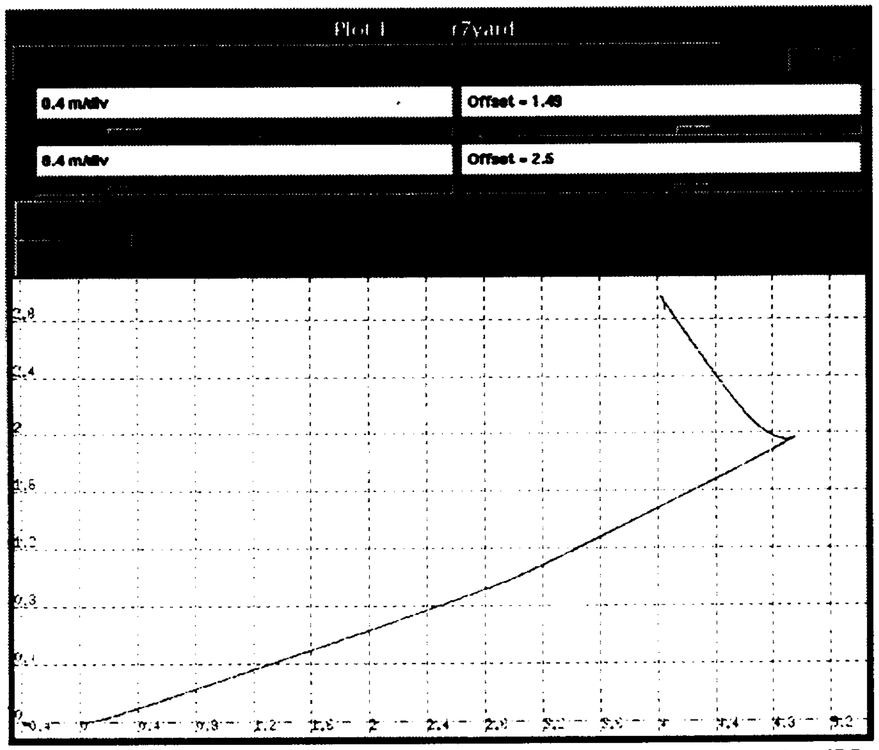

Figure 16. Plot in Real-Time of the Rocky 7 Trajectory on the JPL Mars Yard using Control Shell SetScope Tool

Once the execution of all these path segments are executed autonomously then the rover reports back to the operator that his command was successfully executed. However if the rover cannot find a recovery strategy for a non-nominal situation then the operator will be asked for help and the control operation mode changes in such a way that some control functions switch from software or hardware to operator in the loop.

In detail, the testing of the autonomous piloting layer consists of:

- The ground segment operator generates a sequence of tasks for the camera mast, rover piloting, and robotic arm, and uplinks it via ethernet (simulating DSN).
- An ethernet input control Shell component reads all the received tasks from a VxWork socket and traslates them into Control Shell format (CSMat). Later this ethernet input component introduces all these tasks to the input data queue of the Command Management FSM sending a stimulus to communicate that ground data is ready to process.
- The Command Management FSM interprets the input data, and in this case, send it to the Control Operation Supervisor function (COS), see Fig. 7, that does its own interpretation as well. The Control Operation Supervisor function dispatches a block of commands sequentially to the Mast Control (MC), to the Body Motion Control FSM (BMC), and to the Robotic Arm Control. See Table 4.
- For each path segment, the Body Motion Control FSM will activate several control functions as Control Shell (CS) components that must work in real time. These control functions are:
    - Path segment control consisting of a set of consecutive components for both the piloting nominal feedback (encoder and angular rate sensor reading and position estimation) and piloting forward control functions based on the behavioral approach of having several controllers working in parallel (e.g. a speed and steering controller components) fusioning their outputs (behavior fusion component). See Figures 11 and 12.
    - Body Motion Control will also command to the Wheel Coordination Planning and Wheel Motion Control Layers to activate the inverse kinematic, for Ackerman steering, and wheel motor driver components respectively.
    - Piloting Nominal Feedback of Final Condition consists in a component to detect a cartesian point was reached (see Fig. 12).
    - Piloting Non Nominal Path segment monitoring based on an optical vision system [22] consisting of a set of consecutive components: camera data readers, warpers, Laplacian filters, stereo matching, range and elevation map generators, and obstacle detector (see Fig. 17).
    - Piloting Non Nominal Feedback of Rover attitude monitoring consisting of a set of consecutive components: bogies angle data readers, filter, and dangerous attitude detector.
    - Piloting Non Nominal Feedback of Rover motor monitoring consisting of a set of consecutive components: motor current reading, current filter, and max current detector.
- The Final Condition component will send a message to the BMC nominal feedback queue (see Figures 10 and 12) when the desired cartesian coordinate is reached. Then the BMC will ask for next path segment to COS. If syncronous communiation was selected then COS will send next path segment if any, otherwise the whole path is executed communicating of this fact to the Telemetry Manager FSM that will downlink this event to the ground operator.
- The obstacle and attitude detector, and max current detector will send a message to the non nominal queues of the BMC and WMC FSMs respectively (see Fig. 10 and 12) when its correspondent failure is detected. Hazard recovery strategies will be generated. However if no recovery strategy is possible will downlink this event to the ground operator asking for help.
- Similar control functions to the BMC are activated for the stereo vision mast and robotic arm.

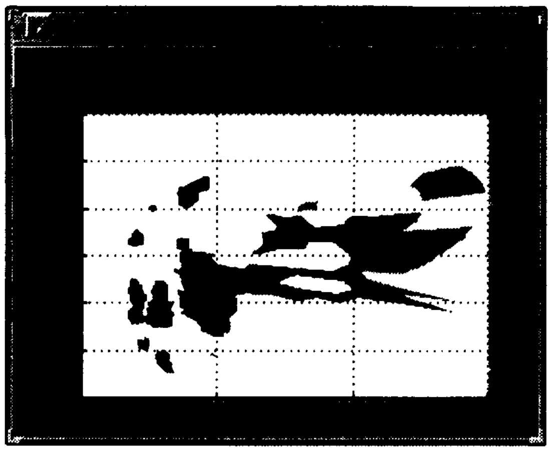
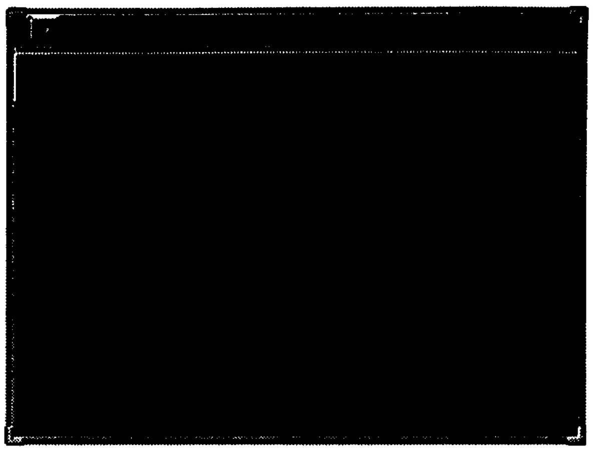

Figure 17. Range Map, Vision Camera Images, and Elevation Map respectively from the Rocky 7 Piloting Non-Nominal Path Segment Monitoring

6. Conclusions and Future Work

A full design of a complete control architecture for a Long Range Science Rover Control System have been proved and the implementation of an autonomous piloting have been shown using successfully the JPL-NASA micro-rover Rocky 7.

A sequence of three Control Architectures (Functional, Operational, and Implementation Control Architectures) have been presented as a good methodology to design a complete control architecture for a complex Space Rover.

An Integrated Control Architecture (ICA) was efficiently shown for inter-element cooperation where the motion control system of each element is based on the Mobile Robot Control Architecture (MORCA) structure.

Space Rover Control Architectures have been successfully shown that they are consistent with MORCA which has extensively been used to define the control architecture for planetary mobile robots, defining a hierarchy of control layers and the internal structure for each layer (Nominal Feedback, Forward Control, and Non Nominal Feedback).

As a gained experience, a pure hierarchical architecture without Non-nominal Feedback (contingency detection and recovery) are the optimum ones. However, due to all the high uncertainties and inaccuracies in a space rover and the mission itself when planning and control are done, non-nominal feedback is needed to take care of the functioning of each control layer and recover the rover from dangerous situations.

A combination of reactive and planning techniques together have been presented for the implementation of the piloting control subsystem to fulfil the severe control requirements of a space mission as to handle high level of uncertainty and inaccuracies.

An autonomous piloting layer has been implemented to successfully achieve in real time continuous driving, detecting and handling non-nominal situations. This real time operation had to deal with the complexity of having numerous and powerful control functions with different operation modes like: cyclic processing of a sequence, parallel execution at convenient sample rates, and event-driven interactions for syncronization, and detection of nominal or non-nominal situations.

All control functions (low and high level) including auxiliary ones for operation support are defined, implemented, and unified in the same graphical tools provided by Control Shell. In addition, Control Shell Real-Time SetScope tool was succesfully used to visualize in real time the Rocky 7 trajectory in its traversy on the JPL Mars Yard during the execution of autonomous piloting.

It was found of big interest the direct correlation existing between MORCA design philosophy and ControlShell graphical tools.

As future work, the implementation of a full autonomous control system is under construction with high level planners and high level generators of recovery strategies. Similar design of Control Architectures are under consideration for other type of autonomous spacecrafts (e.g. landers, chasers, orbiters) therefore other JPL projects, as the ones in the New Millenium Program [9], can benefit of this work as well.

ACKNOWLEDGMENT

Most of the research described in this paper was carried out in Jet Propulsion Laboratory, California Institute of Technology, in the Long Range Science Rover Project under contract with the National Aeronautics and Space Administration. The first author of this paper wants to thank Dr. Samad Hayati and the Spanish Governenment, and more particularly El Ministerio de Educacion y Ciencia Subprograma de Perfeccionamiento de Tecnologos y Doctores en el Extranjero, for his research fellowship that allowed him to continue their research in Space Rovers at JPL.

References

[1] S. Hayati, R. Volpe, et al., "The Rocky 7 Rover: A Mars Sciencecraft Prototype". Proceedings of the IEEE International Conference on Robotics and Automation, Alburquerque NM, April 20-25 1997.

[2] A. Martin-Alvarez, W. De Peuter, P.Putz, "A Unified Control Architecture for Planetary Rovers". Artificial Intelligence, Robotics, and Automation for Space "i-SAIRAS 94", Jet Propulsion Laboratory, Pasadena, California, USA, October 1994.

[3]A. Martin Alvarez, Peter Putz, "An Integrated Architecture for Navigation and Piloting of Cooperating Spacecraft / Rover Systems". 2nd International Symposium Mission, Technologies and Design of Planetary Rovers, Moscow 94.

[4] A. Martin-Alvarez, Maria C. Garcia-Alegre, "Control Architecture for autonomous mobile robots: Linguistic description of navigation and specific manuevering tasks". ISATA 94 (International Symposium on Automative Technology & Automation) Aachen, Germany.

[5] A. Martin-Alvarez. "Piloting and Navigation of Mobile Robots for Planetary Exploration", Doctoral Thesis, Universidad Politecnica de Madrid / Escuela Tecnica Superior de Ingenieros Industriales (in Spanish)), March 1995.

[6] A. Martin-Alvarez, P. Putz, "Interactive Autonomy for Navigation and Piloting of Planetary Rovers", IFAC'96, San Francisco (USA), July 1996.

[7] P. Putz and A. Elfving. "ESA's Control Development Methodology for Space A&R Systems", in Robotics and Manufacturing: Recent Trends in Research, Education, and Applications (M. Jamshidi et al., eds.), Vol. 4, pp. 487-492, ASME Press, 1992.

[8] Stanley A. Schneider, "Extending ControlShell for Intelligent Reactive System". Phase II Final Report. SBIR NASA Contract (NAS7-1391).

[9] Douglas E. Bernard and Barney Pell, "Designed for Autonomy: Remote Agent for the New Millennium Program". ". Artificial Intelligence, Robotics, and Automation for Space "i-SAIRAS '97", Tokyo, Japan, July 1997.

[10] J.S. Albus, "RCS: A reference model architecture for intelligent control",IEEE Journal of Computer Architectures for Intelligent Machines, May 1992.

[11] R. Arkin, "The Impact of Cybernetics on the Design of a Mobile Robot System: A Case Study". IEEE Transactions on Systems, Man, Cybernetics, vol. 20, no.6, November/December 1990.

[12] R. Brooks, "A Robust Layered Control System for a Mobile Robot".IEEE Journal of Robotics ,and Automation, Vol RA.2, No 1,March1986.

[13] E. Gat, M.G. Slack, D.P. Miller, R.J. Firby. "Path planning and execution monitoring for a planetary rover". Proc. of the IEEE Int. Conf. on Robotics and Automation, May 90.

[14] G. Giralt, R. Chatila, M. Vaisset, "An Integrated Navigation and Motion Control System for Autonomous Mobile Robots". In Robotics Research, Ed. by M. Brandy and R.Paul, MIT Press, Cambridge, USA, 1984.

[15] D. Miller, Andrew H. Mishkin, Kenneth E. Lambert, Donald Bickler, Douglas E. Bernard. "Autonomous Navigation and Mobility for a Planetary Rover". AIAA-89-0859.

[16] D. Payton, J.Kenneth Rosenblatt, David M. Keirsey. "Plan Guided Reaction". IEEE Transactions on Systems, Man, and Cybernetics. Vol. 20, No. 6, November/December 1990.

[17] P. Putz, A. Elfving, "A Development Methodology for Space A&R Control Systems". 12th IFAC Symp. on Automatic Control in Aerospace (Aerospace Control 92). Ottobrunn, Sept. 7-11, 1992.

[18] G.N. Saridis. "Control Performance as an Entropy: an Integrated Theory for Intelligent Machines,". Robotics and Automation Laboratory Tech. Rep. No. 19, PRI, Try, New York, 1983.

[19] A. Meystel. "Planning in a hierarchical nested controller for autonomous robots", Proc. IEEE 25th Conf. On Decision and Control, Athens, Greece, 1986.

[20] A. Mishkin, et al, "Experiences with Operations and Autonomy of the Mars Pathfinder Microrover". Proc. Of the 1998 IEEE Aerospace Conference, March 21-28 1998, Snowmass at Aspen, Colorado.

[21] Rocky 7 home page: "http: //robotics.jpl.nasa.gov/tasks/scirover/homepage.html". Maintained by R. Volpe.

[22] L. Matthies and P. Granjean. Stochastic Performance Modeling and Evaluation of Obstacle Detectability with Imaging Range Sensors. IEEE Transaction on Robotics and Automation, 10(6):783-791, December 1994.

[23] R. Volpe, J. Balaram, T, Ohm, and R. Ivlev. The Rocky 7 Mars Rover Prototype. In IEEE//RSJ International Conference on Robots and Systems (IROS), Osaka, Japan, November 4-8 1996.

[24] P. Fortescue, J. Stark. Space Systems Engineering. Wiley Publisher.

[25] J. R. Wetz. Spacecraft Attitude Determination and Control. Kluwer Publisher.

[26] W. L. Larson, J. R. Wertz. Space Mission Analysis and Design. Space Technology Library.

[27] M. H. Kaplan. Modern Spacecraft Dynamics & Control. John Wiley & Sons Publisher.

APPENDICES

APPENDIX 1

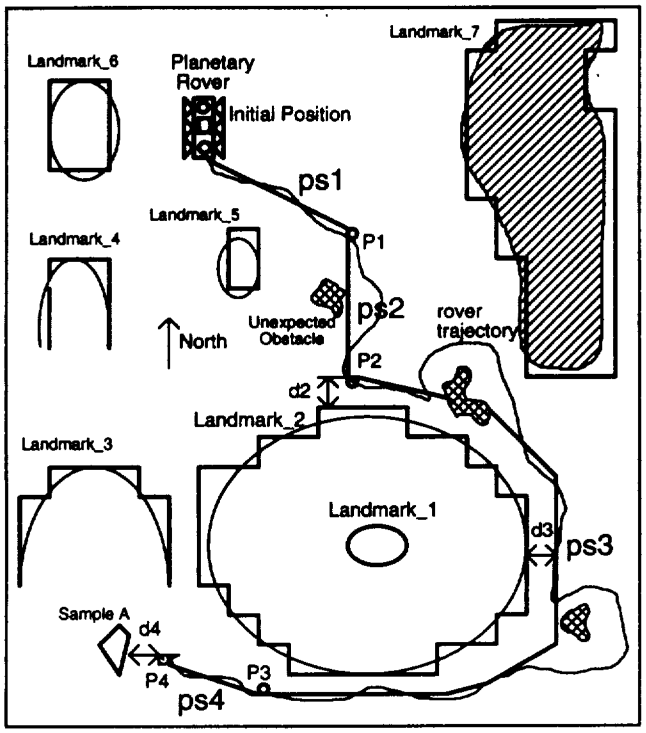

Figure 18. Example of Execution of Mission Cmds

A) Mission Command: "COLLECT_SAMPLE <sample A>"
B) Navigation Command: "GO_TO_LOCATION <sample A>"
C) Piloting Commands:
  C.1) MOVE_TO <Final condition: cardinal point P1>; <Motion reference: cardinal point P1> <Motion direction: forward>
  C.2) REACH <Final condition: cardinal point P2 & external object: landmark_1, in front, distance d2> <Motion reference: external object landmark_1> <Motion direction: forward>
  C.3) BORDER <Final condition: cardinal point P3> <Motion reference: external object: landmark_1, on the right, distance d3> <Motion direction: forward>
  C.4) MOVE_TO <Final condition: cardinal point P4 & external object: sample A, in front, distance d4> <Motion reference: cardinal point P4> <Motion direction: forward>