Increased Flexibility and Robustness of Mars Rovers

John L. Bresina
Phone: 650.604.3365
bresina@ptolemy.arc.nasa.gov

Keith Golden
Phone: 650.604.3585
kgolden@ptolemy.arc.nasa.gov

David E. Smith
Phone: 650.604.4383
de2smith@ptolemy.arc.nasa.gov

Rich Washington*
Phone: 650.604.1140
richw@ptolemy.arc.nasa.gov

Fax: 650.604.3594

NASA Ames Research Center
Mail Stop: 269-2
Moffett Field, CA 94035-1000 USA

ABSTRACT

Our overall objective is to improve the productivity of Mars rovers by increasing the flexibility and robustness of their autonomous behavior. To achieve this objective, we set out to increase the on-board autonomy of rovers and enable commanding at a higher level with a more flexible command language. In February, 1999, we demonstrated some of our rover autonomy technologies as part of a Marsokhod rover field test that simulated aspects of the Mars '01–'05 missions. In this paper, we present the commanding language employed in this field test, called the *Contingent Rover Language (CRL)*, and describe the ground tools and on-board executive capabilities that were developed to generate and execute CRL plans. A key feature of CRL is that it enables the encoding of contingent plans specifying what to do if a failure occurs, as well as what to do if a serendipitous science opportunity arises.

1. INTRODUCTION

Traditionally, spacecraft commanding is accomplished *via* rigid time-stamped sequences of primitive operations. If anything goes wrong during execution, built-in routines attempt to safe the spacecraft and await further instructions from Earth. As NASA missions become more challenging, more sophisticated spacecraft are required, as are more advanced means of commanding them. As a case in point, the Mars Pathfinder's Microrover Flight Experiment made significant advances over previous robotic missions. Sojourner had to operate in an uncertain environment and respond more autonomously to sensor input.

With respect to the Sojourner microrover, for the purposes of this paper, we focus on the issues of commanding and contingency; for more details, see Mishkin, *et al.*, 1998. Like traditional spacecraft, Sojourner was commanded with time-stamped sequences, and the commands tended to be primitive operations. However, there were operations that were specified at a higher level; the primary example is the "Go to Waypoint" command, which implemented autonomous navigation to a specified coordinate.

A command sequence typically specified the activities for one sol (Martian day) plus "runout" commands in case the next sol's sequence was delayed. These sequences contained no explicit contingencies; however, contingency responses to certain drastic scenarios were pre-loaded on both the Pathfinder lander and rover. The "Backup Mission Load" was to be used in the event of a communication loss from Earth to the lander, and the "Contingency Mission Load" was to be used in the event of a communication loss from the lander to the rover.

Our aim is to continue in the technology direction set by the Pathfinder mission and increase the robustness of autonomous rovers by enabling a higher level of commanding with a more flexible and contingent language. The intended benefit is to increase rover productivity without a decrease in safety. Our strategy is to make incremental advancements in this direction so as to maintain relevance to currently planned Mars rover missions and to eventually enable missions beyond the current capabilities of flight rovers.

With planetary rovers, there is uncertainty about many aspects of sequence execution: exactly how long operations will take, how much power will be consumed, and how much data storage will be needed. Furthermore, there is uncertainty about environmental factors that influence such things as rate of battery charging or which scientific tasks are possible. In order to allow for this uncertainty, sequences are typically based on worst-case estimates and contain fail-safe checks. If an operation takes less time than expected, the rover waits for the next time-stamped operation. If operations take longer than expected, they may be terminated before completion. In some cases,

*NASA contractor with Caelum Research Corporation.

all non-essential operations may be halted until a new command plan is received. These situations result in unnecessary delays and lost science opportunities.

Our first steps in this effort involved designing a new commanding language, called the *Contingent Rover Language (CRL)*, described in the next section. A key feature of CRL is that it enables the encoding of contingent plans specifying what to do if a failure occurs, as well as what to do if a serendipitous science opportunity arises. For example, a CRL plan could specify the following contingent rover behavior: when a failure occurs, execute a contingency plan to recover from the failure; if none is available, then execute a contingency plan to acquire additional data to support failure diagnosis and recovery by the ground operations team. We also implemented the ground tools and on-board executive capabilities needed to generate and execute CRL plans, described in the following sections. For further discussion of the ground and on-board techniques, see [Washington, *et al.*, 1999].

In February, 1999, we had an opportunity to demonstrate some of these rover autonomy technologies as part of a field test that was meant to simulate the main objectives of the Mars '01–'05 missions. During this exercise, both advanced rover technologies and science investigation strategies for planetary surface operations were demonstrated. In this paper, we primarily report the aspects of this field test relevant to rover commanding *via* CRL plans.

2. CONTINGENT ROVER LANGUAGE

In this section, we describe a new commanding language, called the *Contingent Rover Language (CRL)*. CRL was designed to serve as the communication medium between the ground operations team and a planetary rover, under the following design criteria.

- **Contingency and Flexibility.** The language should express the constructs that are necessary to achieve scientific goals. In particular, the language should express a variety of temporal and state constraints, and it should support conditional execution of contingency plans based on the execution context.
- **Simplicity.** The language should be simple enough that an automatic, mixed-initiative planning system can provide effective support for plan generation. The intended benefit is to reduce effort on operations staff and to improve the quality of the command plans. Similarly, the language should not be so complex that verification of command plans is impractical. Safety is of paramount importance in space missions, given the high cost of mission failure, so guarantees on execution correctness are critical for any deployed system.
- **Compatibility.** The language should be compatible with existing command languages; *i.e.*, it should allow ground operators to control a rover in the same way that they do now. In particular, it should be possible to easily specify a time-stamped command sequence. The additional capabilities should be available for incremental incorporation as needed to achieve mission goals.

A CRL command plan contains a nominal sequence (possibly) with a set of contingent branches, as well as a library of *alternate plans*. The alternate plans can be thought of as *global contingencies*, whereas the contingent branches are *local contingencies* at specific points in the command plan.

If there are no deviations from the *a priori* execution expectations, then the rover's behavior is governed by the nominal sequence. The contingent branches specify alternative courses of action in response to expectation deviations. Within any contingent branch there may be further contingent branches; hence, the plan is a tree of alternative courses of action.

The alternate plans are not attached to particular points in the command plan; rather, they can be used throughout plan execution, whenever their eligibility conditions are satisfied. When eligible, each alternate plan can either replace the rest of the current plan or be inserted before the rest of the current plan.

Consistent with our *compatibility criterion*, CRL can be used to encode the type of sequences used in the Mars Pathfinder mission, including both the daily uplink sequences as well as the Backup Mission Load and Contingency Mission Load; these loads would be encoded as alternate plans.

Due to our *simplicity criterion*, CRL does not include any control constructs for looping. The design decision we made is that when control loops are needed for execution robustness, they should be embedded within a high-level CRL command. An example of a high-level, robust command with embedded control loops is the "Visual Servo" command, which is somewhat similar to Sojourner's "Go to Waypoint" command. The Visual Servo command, which was used in the 1999 Marsokhod Field Test, implemented autonomous navigation to a specified coordinate *via* visual tracking of a target at that coordinate [Wettergreen, Thomas, and Bualat 1997].

Next, we describe the representations used in CRL. The basic data type in CRL is a *node*. Each node has associated with it a set of conditions that must be satisfied for successful execution; the following are the condition types.

- *start-conditions*: The set of conditions that must be true for the node to begin execution. Conditions can include information about the internal state of the rover (*e.g.*, wheel current), external state (*e.g.*, location), and time windows.
- *wait-for-conditions*: A subset of start-conditions for which the rover will wait until they become true. Other conditions will fail without waiting. Some conditions are automatically waited for whether or not that is specified explicitly; *e.g.*, a constraint on when an action can start executing.
- *maintain-conditions*: A list of conditions that must be true throughout node execution
- *end-conditions*: A list of conditions that must be true at the end of node execution, to verify that an action had the intended effects. Constraints on action duration can be included here.

The conditions can contain variables to be bound during constraint checking; these bindings are used to spe-

cialize the plan according to the execution-time context. The rich expressiveness of temporal and other state constraints on the plan supports effective specification of science goals and safety policies, as well as providing increased flexibility during execution. For example, rather than time-stamps, each action can have a start time interval (and an end time interval).

A node also includes information regarding the expected utility of executing the rest of the plan, as well as information regarding how to react to execution failures: execution may continue to the next node or abort.

CRL has three node subtypes: *block*, *task*, and *branch*; a command plan is defined to be a node, typically of subtype block. A *block* represents a sequence of nodes over which there may be shared state conditions. A *task* represents an action to execute. A task also specifies what action to perform if the task is interrupted due to execution failure. In addition, a task specifies a relative priority and expectations about resource and time usage. A *branch* represents a mutually exclusive choice point in the command plan. Each of the alternative execution paths is represented by an *option*.

An *option* is not a node subtype but a separate data type that has one subtype: *alternate plan*. Options and alternate plans specify the conditions under which they are eligible for execution and the node (typically of subtype block) to execute. In addition to the eligibility conditions, an alternate plan specifies when to check its eligibility: (*i*) whenever a failure occurs, (*ii*) whenever a node finishes execution, or (*iii*) periodically throughout plan execution. As mentioned earlier, when an alternate plan is selected for execution, it can either be inserted before the command plan suffix or it can replace the suffix.

3. CONDITIONAL PLAN EXECUTION

In this section, we describe the version of the on-board executive that was employed in the 1999 Marsokhod Field Test. The conditional executive (CX) is responsible for interpreting the command plan uplinked from ground control, monitoring plan execution, and selecting contingency plans when warranted. CX interacts with the rover control system (RC) and with the Mode Identification system (MI), which performs monitoring and fault diagnosis (described in the next section).

CX starts by executing the nominal sequence of the command plan. At each point in time, CX may have to choose among different courses of action defined by the eligible alternate plans and, if at a branch point, the eligible branch options. CX chooses the course of action with the highest estimated expected utility.

CX receives state information from the Mode Identification system (MI). It uses this information to check the various types of state conditions (in nodes), as well as to check the eligibility conditions of the alternate plans. The ability to branch on any state condition provides the plan writer with a powerful language for specifying rover behavior.

When a failure occurs, CX responds as dictated by the node, either continuing to the next node or aborting the executing plan and checking for eligible alternate plans. In the case that no alternate plans apply, CX aborts the plan and awaits new instructions.

CX communicates with the rover control system (RC) using a datagram model of communication. This communication model allows RC to execute its real-time control loops without blocking on communication, but it carries with it a risk of lost packets. Hence, the communication protocol between CX and RC must be robust to this possibility.

RC broadcasts state and command status information on a continual, periodic basis (currently 10 times/sec). The command status information indicates whether a command is currently executing or terminated; for the latter, success or failure is also indicated.

CX sends out a single packet to initiate action along with a unique command identification. CX then waits for confirmation that RC has received the packet, indicated by seeing a command status (associated with the ID) of executing or terminated. If no such message is received within the time limit, CX will resend the packet. There is a maximum number of command resends that are allowed before causing execution failure. RC ignores the receipt of duplicate command IDs that might arise from the asynchronous communication.

4. MODE IDENTIFICATION

Health maintenance is an important issue for rovers; additionally, in order to support the execution of contingent plans, the executive must have an assessment of the current rover state. The traditional approach for fault detection is to monitor the values of particular sensors and trigger an alarm if a sensor value ever exceeds a given threshold. For example, if the product of current and time ever gets too large (*i.e.*, there is a high current over an extended interval of time), that may indicate a motor stall or other malfunction.

Such a simple mechanism can be useful, but does not easily scale when faults cannot be determined by looking at one or two sensors, or when multiple faults can occur simultaneously. For example, if an ammeter in a motor is failed, then wheel current cannot be used to determine whether the motor has stalled. However, if the encoder (which measures motor position) indicates that the motor is not turning when it should be, that *could* indicate a motor stall. It could also indicate an encoder failure. If other sensors are available, such as accelerometers, cameras, compass or GPS, these could then be used to disambiguate between the two possible failures. Such reasoning is very difficult using the approach discussed above.

Qualitative model-based diagnosis has been successfully applied in such domains, using a model of the system's normal behavior, and optional models of faulty behavior, to produce robust, reliable diagnoses based on all the sensor data, even in the presence of multiple failures. This approach is used in the MIR (Mode-Identification and Reconfiguration) component of the Remote Agent, which flew on board the Deep Space 1 spacecraft [Bernard *et al.*, 1998]. Thus, we decided to use the same system to do mode identification in our architecture. There are many advantages to this approach, which we outline below. However, we also found that due to differences between spacecraft and

rovers, some of the assumptions and design decisions used in MIR are inappropriate for rovers. In the section on the field test experience, we discuss these problems and propose some solutions for them.

The Mode Identification (MI) component of the on-board architecture eavesdrops on commands sent by CX to the rover. As each command is executed, MI receives observations from low-level monitors, which extract qualitative information from the rover sensors. For example, a current monitor may map the continuous-valued current into the set of qualitative values {low, nominal, high}. MI is informed whenever the qualitative value returned by a monitor changes. Based on monitor inputs, the commands executed on the rover, and a declarative model of the rover, MI infers the most likely current state. MI also provides a layer of abstraction to the executive, allowing plans to be specified in terms of component modes, rather than in terms of low-level sensor values.

The behavior of each state of a component is expressed using qualitative, abstract, modular models [Weld and de Kleer, 1990; Williams and de Kleer, 1991], which describe qualities of the rover's structure or behavior without the detail needed for precise numerical prediction. Such models are much easier to acquire and verify than quantitative engineering models, and are easier to reuse. For example, although the Marsokhod has six wheels, each containing a motor, only one wheel module is needed.

While such models cannot specify how far to the left the rover will drift if the motor has failed in one of its left wheels, they can be used to identify the source of failure, given the available sensor data. Such inferences are robust, since small changes in the underlying parameters do not generally affect the high-level behavior of the rover. In addition, abstract models can be reduced to a set of clauses in propositional logic, allowing behavior prediction to use unit propagation, a restricted and very efficient inference procedure.

5. COMMAND PLAN GENERATION

In this section, we discuss the ground tools developed to support the generation of CRL command plans. The process begins with the specification of science goals. CRL was designed to encode not only command plans but also goals. For the 1999 Marsokhod Field Test, a powerful set of intelligent user interface tools was used to support science planning and goal specification. The capabilities provided include generation, display, and manipulation of 3D photorealistic VR models of the rover and its environment; this VR user interface could be used to generate science goals. A separate form-based user interface could also be used to generate and edit CRL goals as well as CRL command plans. The user interface tools also provided the capability to generate CRL command plans with the support of a mixed-initiative, contingent planner/scheduler, which we refer to as *CPS*. For more details on these user interface tools, see Blackmon, *et al.*, 1999 (in this volume).

A typical field test planning cycle proceeded as follows.

1. The scientists provided a set of high-level tasks to be performed on the next simulated sol.
2. Based on this information, we developed a set of high-level CRL tasks using the VR environment and the form-based interface. The VR environment was used for the following tasks: (*i*) to select the best route for drive operations; (*ii*) to help compute angles and distances to targets; and (*iii*) to envision possible obstructions and illumination for image and spectrometer commands.
3. The resulting set of high-level CRL tasks was then passed from the form-based interface to CPS to be recursively decomposed into lower-level tasks and sequences of rover operations. Some decompositions included checks and contingent branches to deal with common faults. In some cases, the decompositions resulted in hundreds of individual rover commands (*e.g.*, panoramic image). If the resulting tasks were unordered, CPS would determine an ordering that satisfied the given time and power constraints.
4. The resulting schedule was passed back to the form-based interface, where it could be displayed and edited. Using the editor, individual steps, groups of steps, or whole branches could be removed or replaced. The resulting schedule fragment was fed back through CPS for any necessary decomposition and completion.
5. Finally, the schedule would be run through a simple syntax checker and uplinked to the on-board rover executive.

In order to allow the kind of mixed-initiative scheduling outlined above, CPS uses a greedy local search strategy. It accepts a seed schedule (possibly empty) and recursively attempts to improve it by fitting additional tasks into gaps in the schedule. When a plateau is reached, tasks already present in the schedule can be exchanged, removed, or shifted. Random walk and restarts further help CPS escape from local minima.

CPS also has the ability to automatically add contingent branches to schedules where appropriate. Building contingency plans is, in general, intractable, and so contingency planners tend to be slow [Draper, Hanks, and Weld, 1994; Pryor and Collins, 1996; Weld, Anderson, and Smith, 1998]. To overcome this problem, CPS employs the *Just-in-Case (JIC)* approach [Drummond, Bresina, and Swanson, 1994], originally developed to handle action duration uncertainty in telescope observation schedules. For the rover domain, we extended the JIC approach as follows.

- To consider uncertainty in power consumption and data production (as well as in task duration).
- To choose among potential contingency branch points based on an assessment of expected utility rather than just probability of failure.
- To allow insertion of setup steps for a contingent branch prior to the actual branch point.

6. THE FIELD TEST EXPERIENCE

In this section, we describe results and lessons learned from our Marsokhod field test experience. The 1999 field test was meant to simulate the main objectives of the Mars '01–'05 missions; the field test employed the

Figure 1: Marsokhod at the 1999 Mojave Field Test.

Ames Marsokhod rover (Figure 1) and took place during February. The remote site was at Silver Lake dry lake bed in California's Mojave desert, and the operations center was at NASA Ames. The field test team consisted of computer scientists and engineers from the NASA Ames Computational Sciences Division, scientists from NASA Ames Space Sciences Division, and planetary scientists from around the world; there were about seventy people who participated.

The Marsokhod platform has been demonstrated at field tests starting with Russian tests in 1993, followed by tests in the Mojave desert in 1994, at Kilauea in Hawaii in 1995, and in the Arizona desert in 1996. Marsokhod is a medium-sized planetary rover built on a Russian chassis. The rover has six wheels, independently driven, with three chassis segments that articulate independently. It is currently configured with imaging cameras that correspond to those planned for use in near-term missions, a spectrometer, and an arm equipped with cameras. The on-board computing environment is a Pentium-based Linux system, for ease of research software integration.

In the rest of this section, we describe the field test results and lessons learned for each of the major autonomy architecture modules: plan execution, mode identification, and plan generation.

6.1. PLAN EXECUTION RESULTS

This was the first Ames field test during which the rover was commanded by uplinking sequences, which were automatically executed on-board, rather than by "joysticking" with the Ames *Virtual Dashboard* interface [Wettergreen, *et al.*, 1997]. A major result of the field test was to build confidence in sequence-based commanding using the CRL language. Although, as expected, complex positioning tasks remain easier through real-time feedback and "joystick" controls, many tasks that involve repetitive activities or precise orientations can be more easily specified and more efficiently executed using CRL.

The following are some examples of how contingency plans were used in the 1999 Marsokhod Field Test and the preparatory readiness tests.

- If a visual-servo command terminates with failure, then acquire an image mosaic to enable relocalization by the operations team.
- If a wheel failure is detected, then acquire images of the failed wheel to support diagnosis.
- If orientation (taco angles) limits are exceeded, then stop and acquire images around all six wheels to support recovery planning.
- During a dead-reckoning traversal, if time (and data storage) allows, then take additional images, to support science and future operations, whenever the rover turns.

Another use of contingent plans is to support on-board, automated science analysis techniques, such as those being developed within the "Graduate Student on Mars (GSOM)" project [Gulick, *et al.*, 1999]. One of the GSOM suite of tools identifies rocks in an image. The following is an example employing this rock-finding algorithm within a contingent command plan. The rover drives a pre-set pattern (*e.g.*, a rectangular circuit) while scanning the environment for rocks. When a rock is found, the rover takes a high-resolution image of the region where GSOM indicates, and it stores this image for later downlink. Other tests, such as spectrometer readings, could be performed on the target location as well, potentially leading to other opportunities for on-board science analysis, *e.g.*, automatically identify carbonates from spectrometer readings. The rover is given a time limit to drive the search pattern, so if it spends too much time analyzing the images and performing tests, it skips some analyses in favor of reaching its way points on schedule.

An important part of robust, autonomous execution is to handle and react to failures that are not within the plan but throughout the system. We have taken steps in that direction with our explicit communication protocol to handle lost packets; however, other challenges remain, such as software failures within real-time controllers or hardware failures in the rover itself. Some of these are handled *via* fault identification by MI and recovery by contingency plans. Some system failures need to be handled in a more comprehensive manner to ensure the rover performs as desired. In particular, approaches ranging from simple heartbeat monitoring and pstate parameter recording to system reconfiguration need to be considered.

The plans constructed by CPS can include contingent branches to handle deviations from expected resource usage. The resources currently considered, in addition to time, are power and data storage. CX could make use of a resource manager to track resource usage and availability, as well as to signal resource conflicts or opportunities. We have developed a prototype resource manager and are integrating it into the on-board executive architecture. The resource manager will ensure that the rover executes its plans within the limits of the available resources and will support branching on a richer set of resource availability conditions.

6.2. MODE IDENTIFICATION RESULTS

Despite the advantages of our approach to state assessment and fault diagnosis, discussed above, MI does have some representation limitations, with respect to modeling rovers. These limitations can be classified as quantitative, probabilistic, and temporal.

Quantitative: There are many advantages to using qualitative models, as outlined above, but many of the more complex aspects of a rover that we would like to model, such as motor behavior and kinematics, are inherently quantitative. Consider again the simple threshold test discussed at the beginning of Section 4: a motor stall is indicated when the current-time product is too high. But how do we determine what is too high? "Normal" wheel currents depend on whether the rover is turning, driving uphill, or going over rocky terrain. The expected current, thus, is a quantitative function of factors such as pitch, turning, and bumpiness (as measured, perhaps, by accelerometers).

The approach we have taken is to use a purely qualitative model, abstracting away quantitative details using monitors. However, using this approach, we either end up with most of the complexity hidden in the monitors, or we are forced to discretize the values in question into many intervals and rely on qualitative arithmetic to do the math in MI, which can be computationally expensive. It would be much simpler and more efficient to work with the numbers directly. This is possible by incorporating quantitative models, using hybrid continuous/discrete systems, such as HCC [Carlson and Gupta, 1998]. HCC is already used for a simulation of the Marsokhod [Sweet, Blackmon, and Gupta, 1999], and work is underway to combine it with MI, for use in diagnosis. We are also considering the use of Kalman filters, which are ideal for combining numerical data from multiple noisy sensors, and which have successfully been used in MIR monitors.

Probabilistic: In MI, transitions to particular states can be conditional or probabilistic, but not both. That is, they are ether deterministic, commanded transitions into "okay" modes, or unconditional random transitions into fault modes. Many aspects of the rover behavior involve conditional probabilistic transitions. For example, going up a steep hill results in high torque on the rear wheels, which leads to an increased probability that the wheel motors will stall.

With the current representation, we cannot express the fact that motor stalls are more likely to occur in the presence of high torques. To do so, we need conditional probabilities. Effectively using conditional probabilities requires tracking multiple trajectories, which is not currently done in MI for efficiency reasons; thus, entailing a larger computational burden. We are also considering other representations, including Markov decision processes.

Temporal: One of the assumptions underlying MI is that the system being monitored is synchronos, spending most of its time in a steady state (at least at the qualitative level reflected by the models) and that transitions between states are rapid enough that by simply waiting for quiescence, MI can treat them as instantaneous. However, on the rover, this assumption is violated. State transitions are sufficiently frequent and transitions between states are sufficiently slow that there is no guarantee that the rover will reach a steady state. This is due in part to a high degree of uncertainty in the time that will be required for a transition to occur.

6.3. PLAN GENERATION RESULTS

During the field test, we learned a number of lessons about generating command plans. Some parts of the process worked very well, but there were places where we clearly needed additional software tools, or needed to improve the capabilities of our existing tools.

Probably the most glaring omission was the lack of adequate tools to allow the scientists to generate high-level CRL tasks directly. Although a web interface was developed for this purpose, it did not cover the full spectrum of possible scientific experiments and objectives. In addition, the interface did not allow them to specify temporal constraints and did not provide adequate feedback concerning resource requirements or expected data production for proposed experiments. As a result, the interface received little use by the scientists and, instead, the scientific goals were relayed verbally. As a result, significant manual labor was involved in turning the scientists requests into a fleshed out set of high-level CRL tasks.

In contrast, we made extensive use of automated decomposition of high-level science tasks into detailed sequences of rover commands. This capability was essential for efficient development of command plans. In some cases, the command plans contained hundreds of commands and we simply could not have generated these by hand in the time allotted.

We did not make significant use of the automated scheduling capabilities. The primary reason for this is that for each sol the scientists were providing a specific ordered set of tasks to be performed. They did not provide a larger set of prioritized tasks from which choices could be made, based on time, power, and data considerations. This was due, at least in part, to the relatively short duration of the field test, which did not allow the scientists to develop a set of longer-term objectives. Additionally, the scientists were not made aware of how to take full advantage of the capabilities CPS could provide. For an extended mission with a larger number of distributed scientists submitting requests, we believe that the scheduling capability would become more important, especially if employed to generate multi-sol command plans.

We also did not make significant use of automatically generated contingency branches. Without a larger set of tasks to choose from, CPS cannot build useful alternative branches. However, even with a larger set of tasks, CPS would not have been able to anticipate many of the failures that occured during the field test.

Currently, CPS only develops contingent branches for failures that result from time and resource conflicts. During the field test, most of the plan failures were due to other things, such as losing visual targets during traverses and motor current anomalies. In these cases, useful alternative plans could have been developed automatically, but to do so, we need to enrich the set of potential failures considered by CPS.

7. CONCLUDING REMARKS

In this paper, we presented the *Contingent Rover Language (CRL)* for commanding planetary rovers, and we described the ground-based and on-board systems that were demonstrated in the 1999 Marsokhod Mojave Field Test. Our overall objective is to increase the flexibility and robustness of autonomous rover behavior in order to improve science productivity. The initial efforts towards this objective (reported here) focused on the concept of "contingency". CRL allows the specification of contingent courses of action for the purposes of recovering from expectation failures or taking advantage of serendipitous science opportunity. Our mixed-initiative planner/scheduler (CPS) supports the generation of contingent CRL command plans and our on-board executive systems (CX and MI) enable robust plan execution that is responsive to the runtime, dynamic environment.

In the previous section, we mentioned future work directions for each of the three component technologies. In addition, we intend to pursue command plan verification. In order to support verification, as well as plan generation, we plan to integrate rover simulation with constraint reasoning and planning techniques. In the future, we would also like to migrate some of the planning activities on-board the rover as appropriate; for example, the ability to replan science activities in response to on-board science analysis and runtime conditions (*e.g.*, resource availability).

ACKNOWLEDGMENTS

We'd like to acknowledge the following collaborators who helped with the work reported here: Corin Anderson for his help with CPS and user interface; Katherine Smith for her help with the MI system and Marsokhod models; Trey Smith for his help with plan execution; and Corin Anderson, Ted Blackmon, David Miller, Barney Pell, and Trey Smith for their help in CRL development. The implementation of CX uses constructs from ESL [Gat, 1996].

We'd also like to acknowledge the following collaborators that worked on Marsokhod's hardware, onboard software, user interfaces, stereo modeling, simulation, and other ground tools: Ted Blackmon, Maria Bualat, Vineet Gupta, Gary Haith, Aaron Kline, Linda Kobayashi, Cesar Mina, Charles Neveu, Laurent Nguyen, Sergey Sokolov, Hans Thomas, Anne Wright, and Eric Zbinden.

We'd like to thank all the field test collaborators: John Schreiner (Field Test Lead), Michael Sims (Technical Lead), Hans Thomas (Engineering Lead), Carol Stoker (Science Lead), and Nathalie Cabrol (Science Deputy), the NIR and MIR instrument teams, the GSOM team, and the other 50+ participants. We'd also like to thank NASA Ames management, in particular within the Computational Sciences Division, for supporting our research and the field test activity.

REFERENCES

Bernard, D.E., Dorais, G.A., Fry, C., Gamble Jr., E.B., Kanefsky, R., Kurien, J., Millar, W., Muscettola, N., Nayak, P.P., Pell, B., Rajan, K., Rouquette, N., Smith, B., and Williams, B.C. 1998. Design of the remote agent experiment for spacecraft autonomy. In *Proc. of the IEEE Aerospace Conference.*

Blackmon, T., *et al.*, 1999. Command generation for planetary rovers using virtual reality. In *Proc. of i-SAIRAS.*

Carlson, B., and Gupta, V. 1998. Hybrid CC and interval constraints. In *Proc. of the International Workshop on Hybrid Systems: Computation and Control.*

Draper, D., Hanks, S., and Weld, D. 1994. Probabilistic planning with information gathering and contingent execution. *Proc. of the Second International Conference on AI Planning Systems.*

Drummond, M., Bresina, J., and Swanson, K. 1994. Just-in-case scheduling. *Proc. of the Twelfth National Conference on AI.*

Gat, E. 1996. ESL: A language for supporting robust plan execution in embedded autonomous agents. *Proc. of the AAAI Fall Symposium on Plan Execution.*

Gulick, V.C., Morris, R.L. , Ruzon, M., and Roush, T.L. 1999. Autonomous science analysis of digital images for Mars sample return and beyond. *The 30th Lunar Planetary Science Conference.*

Mishkin, A.H., Morrison, J.C., Nguyen, T.T., Stone, H.W., Cooper, B.K., and Wilcox, B.H. 1998. Experiences with Operations and Autonomy of the Mars Pathfinder Microrover. *Proc. of the IEEE Aerospace Conference.*

Pryor, L., and Collins, G. 1996. Planning for contingencies: A decision-based approach. *Journal of AI Research.*

Sweet, A., Blackmon, T., and Gupta, V. 1999. Simulation of a Rover and Display in a Virtual Environment. In *Proc. of the American Nuclear Society 8th International Topical Meeting on Robotics and Remote Systems.*

Washington, R., Golden, K., Bresina, J., Smith, D.E., Anderson, C., and Smith, T. 1999. Autonomous rovers for Mars exploration. In *Proc. of the 1999 IEEE Aerospace Conference.*

Weld, D.S. and de Kleer, J. 1990. *Readings in Qualitative Reasoning About Physical Systems*, Morgan Kaufmann Publishers, Inc., San Mateo, California.

Weld, D.S., Anderson, C.R., and Smith, D.E. 1998. Extending Graphplan to handle uncertainty and sensing actions. *Proc. of the Fifteenth National Conference on AI.*

Wettergreen, D., Bualat, M., Christian, D., Schwehr, K., Thomas, H. Tucker, D., and Zbinden, E. 1997. Operating Nomad during the Atacama Desert Trek. In *Proc. Field and Service Robotics Conference.*

Wettergreen, D., Thomas, H., Bualat, M. 1997. Initial Results from Vision-based Control of the Ames Marsokhod Rover. In *Proc. of the International Conference on Intelligent Robots and Systems, Control of Wheeled Robots.*

Williams, B. C. and de Kleer, J. 1991. Qualitative reasoning about physical systems: A return to roots. *Artificial Intelligence*, 51:1-10.